**HW6**

**LAB 8**

**Name: Abdurasulov Mirsaid**

**ID: 12225253**

**Field: ISE**

**Course: Object oriented programming**

**Date: 31.10.2023**

Exercise #1:

Source code:

#include <iostream>

using namespace std;

class Calculate {

private:

double y = 0; // Initial value

public:

// Get the current value

double getValue() {

return y;

}

// Perform addition and update the value

double add(double value) {

y = y + value;

return y;

}

// Perform subtraction and update the value

double subtract(double value) {

y = y - value;

return y;

}

// Perform multiplication and update the value

double multiply(double value) {

y = y \* value;

return y;

}

// Perform division and update the value

double divide(double value) {

y = y / value;

return y;

}

};

int main(void) {

Calculate cc;

int i;

char op;

double value;

// Loop for performing mathematical operations

for (i = 0; i < 10; ++i) {

cout << "Select math operator(+,-,\*,/): ";

cin >> op;

cout << "Enter a real number for the math: ";

cin >> value;

// Switch statement to choose the operation based on the operator entered

switch (op) {

case '+':

cout << cc.getValue() << " + " << value;

cout << " = " << cc.add(value) << endl;

break;

case '-':

cout << cc.getValue() << " - " << value;

cout << " = " << cc.subtract(value) << endl;

break;

case '\*':

cout << cc.getValue() << " \* " << value;

cout << " = " << cc.multiply(value) << endl;

break;

case '/':

cout << cc.getValue() << " / " << value;

cout << " = " << cc.divide(value) << endl;

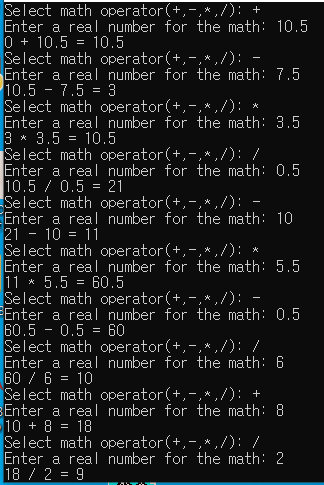
break;

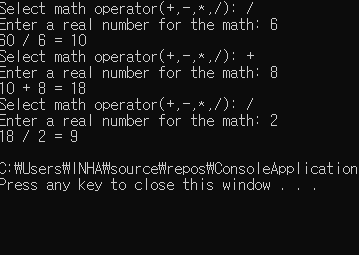
}

}

}

**Result:**





**Exercise #2:**

**Source code:**

// Abdurasulov Mirsaid 12225253

#include <iostream>

using namespace std;

class Calculate {

private:

double value = 0.0; // default variable with the value of 0.0 inside private

char operators[5];

double s\_values[5];

int index = -1;

public:

// function which returns the value of the value variable

double getValue() {

return value;

}

double add(double a) { // add function which adds the real number to value

store\_operarion('+', a);

value += a;

return value;

}

double substract(double a) { // substract function which - the real number from value

store\_operarion('-', a);

value -= a;

return value;

}

double multiply(double a) { // multiply function which multiplies the real number to the value

store\_operarion('\*', a);

value \*= a;

return value;

}

double divide(double a) { // devide function which devides the real number to the value

store\_operarion('/', a);

value /= a;

return value;

}

void store\_operarion(char op, double svalue) {

if (index < 4) {

index++;

operators[index] = op;

s\_values[index] = svalue;

}

else {

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

operators[i] = operators[i + 1];

s\_values[i] = s\_values[i + 1];

}

operators[4] = op;

s\_values[4] = svalue;

}

}

bool lastOperation(char& op, double& svalue) {

if (index >= 0) {

op = operators[index];

svalue = s\_values[index];

return true;

}

return false;

}

void undo() {

if (index >= 0) {

char op = operators[index];

double svalue = s\_values[index];

index--;

if (op == '+') value -= svalue;

else if (op == '-') value += svalue;

else if (op == '\*') value /= svalue;

else if (op == '/') value \*= svalue;

}

}

};

int main(void) {

Calculate cc;

int i;

char op;

double value;

for (i = 0; i < 10; ++i) {

cout << "Select math operator(+,-,\*,/): ";

cin >> op;

cout << "Enter a real number for the math: ";

cin >> value;

switch (op) {

case '+':

cout << cc.getValue() << " + " << value;

cout << " = " << cc.add(value) << endl;

break;

case '-':

cout << cc.getValue() << " - " << value;

cout << " = " << cc.substract(value) << endl;

break;

case '\*':

cout << cc.getValue() << " \* " << value;

cout << " = " << cc.multiply(value) << endl;

break;

case '/':

cout << cc.getValue() << " / " << value;

cout << " = " << cc.divide(value) << endl;

break;

}

}

cout << endl << endl;

for (i = 0; i < 10; ++i) {

cout << "Stored the last math operation: ";

bool flag = cc.lastOperation(op, value);

if (!flag)

cout << "None" << endl;

else

cout << op << ", " << value << endl;

if (flag) {

cc.undo();

cout << "Undo the last math operation..." << endl;

cout << "Value inside the class object: ";

cout << cc.getValue() << endl;

}

}

cout << "Value inside the class object: ";

cout << cc.getValue() << endl;

return 0;

}

**Results:**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Exercise #3:**

**Source code:**

#include <iostream>

using namespace std;

class Calculate {

public:

double a = 0;

char lastOp = '\0'; // Store the last operation.

double lastValue = 0; // Store the last value.

Calculate() = default; // Default constructor

Calculate(const Calculate& other) {

a = other.a; // Copy the final result

lastOp = '\0'; // Clear the past operation history

lastValue = 0;

}

double getValue() {

return a;

}

double add(double value) {

a = a + value;

setLastOperation('+', value);

return a;

}

double subtract(double value) {

a = a - value;

setLastOperation('-', value);

return a;

}

double multiply(double value) {

a = a \* value;

setLastOperation('\*', value);

return a;

}

double divide(double value) {

a = a / value;

setLastOperation('/', value);

return a;

}

void setLastOperation(char op, double value) {

lastOp = op;

lastValue = value;

}

bool lastOperation(char& op, double& value) {

if (lastOp == '\0') {

return false;

}

op = lastOp;

value = lastValue;

return true;

}

void undo() {

if (lastOp == '+') {

a -= lastValue;

}

else if (lastOp == '-') {

a += lastValue;

}

else if (lastOp == '\*') {

a /= lastValue;

}

else if (lastOp == '/') {

a \*= lastValue;

}

lastOp = '\0'; // Clear the last operation after undoing.

}

};

int main(void) {

Calculate cc;

int i;

char op;

double value;

bool flag;

for (i = 0; i < 10; ++i) {

cout << "Select math operator(+,-,\*,/): ";

cin >> op;

cout << "Enter a real number for the math: ";

cin >> value;

switch (op) {

case '+':

cout << cc.getValue() << " + " << value;

cout << " = " << cc.add(value) << endl;

break;

case '-':

cout << cc.getValue() << " - " << value;

cout << " = " << cc.subtract(value) << endl;

break;

case '\*':

cout << cc.getValue() << " \* " << value;

cout << " = " << cc.multiply(value) << endl;

break;

case '/':

cout << cc.getValue() << " / " << value;

cout << " = " << cc.divide(value) << endl;

break;

}

}

cout << endl << endl;

Calculate dd{ cc }; // Create a copy of the Calculate object

cout << endl << endl;

for (i = 0; i < 10; ++i) {

cout << "Stored the last math operation: ";

flag = dd.lastOperation(op, value);

if (!flag)

cout << "None" << endl;

else

cout << op << ", " << value << endl;

if (flag) {

dd.undo();

cout << "Undo the last math operation..." << endl;

cout << "Value inside the class object: ";

cout << dd.getValue() << endl;

}

}

cout << "Value inside the class object: ";

cout << dd.getValue() << endl;

return 0;

}

**Results:**

A screenshot of a computer

Description automatically generated

**Exercise #4:**

**Source code:**

#include <iostream>

#include <iomanip>

using namespace std;

class Time {

private:

int hour;

int min;

int sec;

public:

// Constructor to initialize Time object

Time(int h, int m, int s) : hour(h), min(m), sec(s) {};

// Print time in 12-hour format

void printTime() const {

if (hour <= 23 && hour > 12) {

// If the hour is in the afternoon/evening, subtract 12 to display in 12-hour format

cout << setfill('0') << setw(2) << hour - 12 << ":";

cout << setfill('0') << setw(2) << min << ":";

cout << setfill('0') << setw(2) << sec << endl;

}

else {

// For morning hours (before 12 PM), display time as is

cout << setfill('0') << setw(2) << hour << ":";

cout << setfill('0') << setw(2) << min << ":";

cout << setfill('0') << setw(2) << sec << endl;

}

}

// Non-const version of printTime for the Time object

void printTime() {

// Display time in 24-hour format (no conversion)

cout << setfill('0') << setw(2) << hour << ":";

cout << setfill('0') << setw(2) << min << ":";

cout << setfill('0') << setw(2) << sec << endl;

}

};

int main() {

const Time t0{ 7, 3, 5 }; // Constant Time object t0

Time t1{ 13, 45, 9 }; // Non-constant Time object t1

const Time t2{ t1 }; // Constant Time object t2 initialized from t1

cout << "t0: ";

t0.printTime(); // Display t0 time in 12-hour format

cout << "t1: ";

t1.printTime(); // Display t1 time in 24-hour format

cout << "t2: ";

t2.printTime(); // Display t2 time in 12-hour format

return 0;

}

**Result:**

A screen shot of a digital clock

Description automatically generated

**Exercise #5:**

**Source code:**

#include <iostream>

#include <iomanip>

#include <ctime>

#include <cstdlib>

using namespace std;

const int arSize = 20;

class IntArray {

private:

int m\_len{ 0 };

int\* m\_data{ nullptr };

public:

IntArray(int len)

: m\_len{ len }

{

m\_data = new int[m\_len];

}

// Friend functions

friend void setArray(IntArray& ar, int date[], const int arSize);

friend void displayArray(IntArray& ar);

~IntArray() {

if (m\_data) delete[] m\_data;

}

};

// Function to set the contents of the IntArray using an array

void setArray(IntArray& ar, int date[], int arSize) {

ar.m\_len = arSize; // Update the length

// Allocate new memory for the array and copy the data from the input array

ar.m\_data = new int[ar.m\_len];

for (int i = 0; i < ar.m\_len; ++i) {

ar.m\_data[i] = date[i];

}

}

// Function to display the contents of the IntArray

void displayArray(IntArray& ar) {

for (int i = 0; i < ar.m\_len; i++) {

cout << "[" << i + 1 << "] " << ar.m\_data[i] << endl;

}

}

int main() {

int i;

int data[arSize];

IntArray ar{ arSize };

srand(static\_cast<unsigned int>(time(NULL)));

for (i = 0; i < arSize; ++i) {

data[i] = rand() % 100;

}

setArray(ar, data, arSize);

displayArray(ar);

return 0;

}

**Result:**

A screenshot of a black screen

Description automatically generated