Course: ENSF 614 – Fall 2023

Lab 1:

Instructor: M. Moussavi

Student Name: Emmanuel Alafonye

Submission Date: October 13, 2023

/\*

\* File Name: lab3exe\_A.cpp

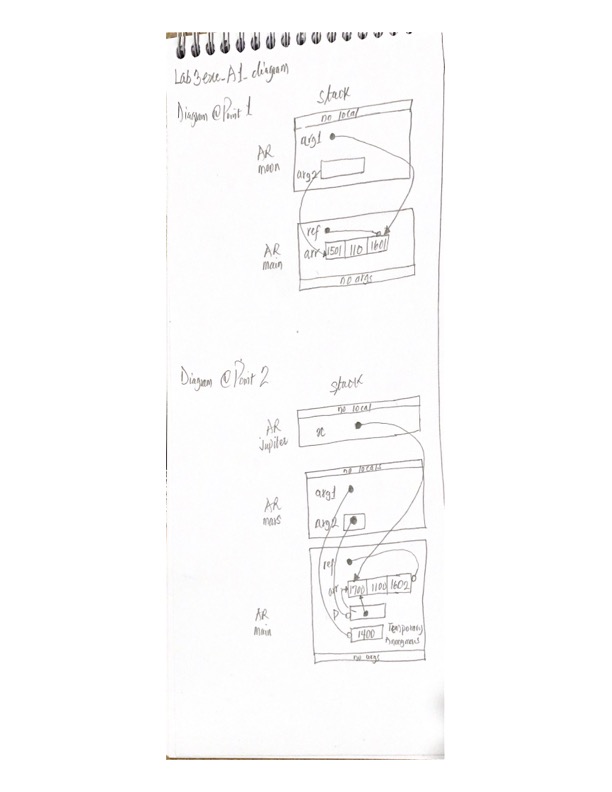
\* Assignment: ENSF 614 Lab 3, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 13, 2023.

\*/



A whiteboard with a drawing of a computer

Description automatically generated with medium confidence

/\*

\* File Name: lab3exe\_B.cpp

\* Assignment: ENSF 614 Lab 3, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 13, 2023.

\*/

A white board with a drawing on it

Description automatically generated

Lab3Clock.h

/\*

\* File Name: lab3exe\_C.cpp

\* Assignment: ENSF 614 Lab 3, exercise C

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 13, 2023.

\*/

#ifndef LAB3CLOCK\_H

#define LAB3CLOCK\_H

// Preprocessor directive that prevents multiple inclusions of this header file.

// If LAB3CLOCK\_H is not defined, the code between #ifndef and #endif will be included.

class Clock {

public: // Public member function to get components

Clock(); // Default constructor

Clock(int seconds); // Constructor with seconds argument

Clock(int hours, int minutes, int seconds); // Constructor with hours, minutes, and seconds

int get\_hour() const;

int get\_minute() const;

int get\_second() const;

void set\_hour(int hours);

void set\_minute(int minutes);

void set\_second(int seconds);

void increment();

void decrement();

void add\_seconds(int seconds);

private: // Private data to store components

int hour;

int minute;

int second;

int hms\_to\_sec() const;

void sec\_to\_hms(int total\_seconds);

};

#endif

Lab3Clock.cpp

#include "lab3Clock.h"

// Default constructor

Clock::Clock() {

hour = 0;

minute = 0;

second = 0;

}

// Constructor with seconds argument

Clock::Clock(int seconds) {

sec\_to\_hms(seconds);

}

// Constructor with hours, minutes, and seconds

Clock::Clock(int hours, int minutes, int seconds) {

if (hours >= 0 && hours <= 23 && minutes >= 0 && minutes <= 59 && seconds >= 0 && seconds <= 59) {

hour = hours;

minute = minutes;

second = seconds;

} else {

hour = 0;

minute = 0;

second = 0;

}

}

// Getter functions

int Clock::get\_hour() const {

return hour;

}

int Clock::get\_minute() const {

return minute;

}

int Clock::get\_second() const {

return second;

}

// Setter functions

void Clock::set\_hour(int hours) {

if (hours >= 0 && hours <= 23) {

hour = hours;

}

}

void Clock::set\_minute(int minutes) {

if (minutes >= 0 && minutes <= 59) {

minute = minutes;

}

}

void Clock::set\_second(int seconds) {

if (seconds >= 0 && seconds <= 59) {

second = seconds;

}

}

// Additional functionalities

void Clock::increment() {

if (hour == 23 && minute == 59 && second == 59) {

hour = 0;

minute = 0;

second = 0;

} else {

add\_seconds(1);

}

}

void Clock::decrement() {

if (hour == 0 && minute == 0 && second == 0) {

hour = 23;

minute = 59;

second = 59;

} else {

add\_seconds(-1);

}

}

void Clock::add\_seconds(int seconds) {

int total\_seconds = hms\_to\_sec() + seconds;

sec\_to\_hms(total\_seconds);

}

// Helper functions

int Clock::hms\_to\_sec() const {

return hour \* 3600 + minute \* 60 + second;

}

void Clock::sec\_to\_hms(int total\_seconds) {

if (total\_seconds >= 0 && total\_seconds <= 86399) {

hour = total\_seconds / 3600;

minute = (total\_seconds % 3600) / 60;

second = total\_seconds % 60;

} else {

hour = 0;

minute = 0;

second = 0;

}

}

A screenshot of a computer program

Description automatically generated

/\*

\* File Name: lab3exe\_D.cpp

\* Assignment: ENSF 614 Lab 3, exercise D

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* Submission Date: October 13, 2023.

\*/

#include "MyArray.h"

MyArray::MyArray()

{

sizeM = 0; // Creating the empty array.

storageM = new EType[0]; // set the size

}

// Constructor for the object array.

MyArray::MyArray(const EType \*builtin, int sizeA){

sizeM = sizeA;

storageM = new EType[sizeM];

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

storageM[i] = builtin[i];

}

}

// Copy constructor for the myArray class

MyArray::MyArray(const MyArray &source){

sizeM = source.sizeM;

storageM = new EType[sizeM];

for (int i = 0; i < sizeM; i++)

{

storageM[i] = source.storageM[i];

}

}

MyArray &MyArray::operator=(const MyArray &rhs){

if (this == &rhs){

return \*this; // Handle self-assignment

}

delete[] storageM;

sizeM = rhs.sizeM;

storageM = new EType[sizeM];

for (int i = 0; i < sizeM; i++)

{

storageM[i] = rhs.storageM[i];

}

return \*this;

}

// Destructor

MyArray::~MyArray(){

delete[] storageM;

}

int MyArray::size() const{

return sizeM;

}

// Returns the element of specified index.

EType MyArray::at(int i) const{

if (i >= 0 && i < sizeM)

{

return storageM[i];

}

else

{

return 0;

}

}

void MyArray::set(int i, EType new\_value)

{

if (i >= 0 && i < sizeM)

{

storageM[i] = new\_value;

}

}

// Resizes the array to a new size

void MyArray::resize(int new\_size){

if (new\_size < 0)

{

return;

}

EType \*newStorage = new EType[new\_size];

int copySize = (new\_size < sizeM) ? new\_size : sizeM;

for (int i = 0; i < copySize; i++)

{

newStorage[i] = storageM[i];

}

delete[] storageM;

storageM = newStorage;

sizeM = new\_size;

}

A screenshot of a computer program

Description automatically generated