МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»**

Институт ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ЦИФРОВЫЕ ТРАНСФОРМАЦИИ

**Отчет по лабораторной работе № 3**

**по дисциплине «Объектно-ориентированное программирование»**

**Тема: «Классы. Инкапсуляция. Реализация класса MyTime»**

Выполнил: Сидоров Д. С., группа ИТС-122

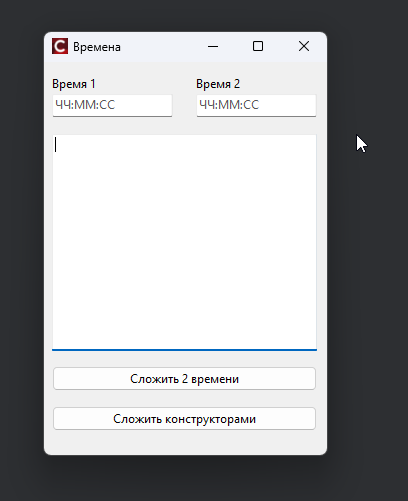
Проверил: к.т.н., доц. Семёнов А. А.

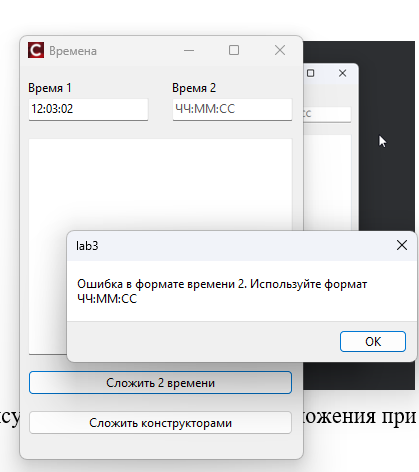
Москва, 2025г

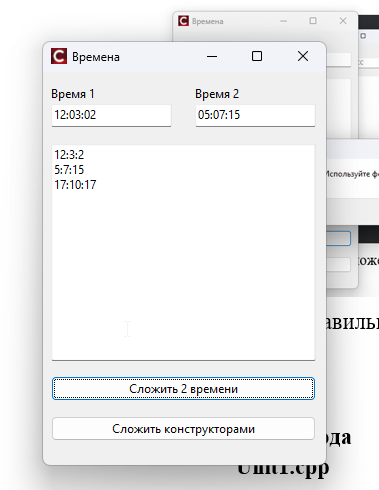
**Классы. Инкапсуляция. Реализация класса MyTime**

Задание: изучить и повторить самостоятельно иллюстрированный пример (см. ниже), демонстрирующий реализацию следующей задачи. Разработать визуальное приложение, в котором необходимо создать класс с именем MyTime, содержащий три поля типа int, предназначенные для хранения часов, минут и секунд. Один из конструкторов класса должен инициализировать поля нулевыми значениями, а другой конструктор – заданным набором значений. Создайте метод класса, который будет выводить значения полей на экран в формате 23:59:59, и метод, складывающий значения двух объектов типа MyTime, передаваемых в качестве аргументов. В обработчике события ButtonClick следует создать два инициализированных объекта и один неинициализированный объект, затем сложить два инициализированных значения, а результат присвоить третьему объекту и вывести его значение на экран (например, 13:23:50 + 10:52:50 = 0:16:40). Повторив рассмотренный в п.1 пример, необходимо обеспечить возможность ввода пользователем значений переменных Т1 и Т2. Для этого необходимо дополнительно разместить на форме два компонента LabeledEdit, которые и будут обеспечивать ввод значений для Т1 и Т2, и реализовать считывание этих значений в обработчике Button1Click посредством вызова соответствующего метода, который считывает введенные пользователем данные. Время пользователь должен вводить по маске ЧЧ:ММ:СС, а также необходимо обеспечить защиту от ввода некорректных данных.

Также необходимо реализовать обработчик Button2Click, который будет использовать вызов конструкторов для инициализации объектов T1 и T2.

  
Рисунок 1. Начальный скрин приложения при открытии

  
Рисунок 2. Ошибка при неправильных данных

  
Рисунок 3. Сложение времён

**Листинг кода**

**Unit1.cpp**

///---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

void MyTime::show(){

AnsiString s;

s = IntToStr(chas) + ":" + IntToStr(min) + ":" + IntToStr(sec);

Form1->Memo1->Lines->Add(s);

}

void MyTime::summa(MyTime t1, MyTime t2){

sec = t1.sec + t2.sec;

min = t1.min + t2.min;

chas = t1.chas + t2.chas;

if(sec>=60) { min++; sec-=60; }

if(min>=60) { chas++; min-=60; }

if(chas>=24) chas = chas-24;

}

bool MyTime::parseTimeString(const AnsiString &timeStr) {

try {

// Проверяем формат ЧЧ:ММ:СС

if (timeStr.Length() != 8 || timeStr[3] != ':' || timeStr[6] != ':')

return false;

chas = StrToInt(timeStr.SubString(1, 2));

min = StrToInt(timeStr.SubString(4, 2));

sec = StrToInt(timeStr.SubString(7, 2));

// Проверка допустимых диапазонов

if (chas < 0 || chas > 23 || min < 0 || min > 59 || sec < 0 || sec > 59)

return false;

return true;

}

catch (...) {

return false; // Ошибка преобразования строки в число

}

}

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

// Устанавливаем подсказку для полей ввода

LabeledEdit1->TextHint = "ЧЧ:ММ:СС";

LabeledEdit2->TextHint = "ЧЧ:ММ:СС";

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

Memo1->Clear();

MyTime T1, T2, T3;

// Проверяем корректность ввода времени 1

if (!T1.parseTimeString(LabeledEdit1->Text)) {

ShowMessage("Ошибка в формате времени 1. Используйте формат ЧЧ:ММ:СС");

LabeledEdit1->SetFocus();

return;

}

// Проверяем корректность ввода времени 2

if (!T2.parseTimeString(LabeledEdit2->Text)) {

ShowMessage("Ошибка в формате времени 2. Используйте формат ЧЧ:ММ:СС");

LabeledEdit2->SetFocus();

return;

}

T1.show();

T2.show();

T3.summa(T1, T2);

T3.show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

Memo1->Clear();

// Парсим введенные пользователем данные для создания объектов

try {

// Разбор строки для T1

AnsiString str1 = LabeledEdit1->Text;

if (str1.Length() != 8 || str1[3] != ':' || str1[6] != ':') {

ShowMessage("Ошибка в формате времени 1. Используйте формат ЧЧ:ММ:СС");

LabeledEdit1->SetFocus();

return;

}

int ch1 = StrToInt(str1.SubString(1, 2));

int min1 = StrToInt(str1.SubString(4, 2));

int sec1 = StrToInt(str1.SubString(7, 2));

// Проверка диапазонов для T1

if (ch1 < 0 || ch1 > 23 || min1 < 0 || min1 > 59 || sec1 < 0 || sec1 > 59) {

ShowMessage("Некорректные значения времени 1");

LabeledEdit1->SetFocus();

return;

}

// Разбор строки для T2

AnsiString str2 = LabeledEdit2->Text;

if (str2.Length() != 8 || str2[3] != ':' || str2[6] != ':') {

ShowMessage("Ошибка в формате времени 2. Используйте формат ЧЧ:ММ:СС");

LabeledEdit2->SetFocus();

return;

}

int ch2 = StrToInt(str2.SubString(1, 2));

int min2 = StrToInt(str2.SubString(4, 2));

int sec2 = StrToInt(str2.SubString(7, 2));

// Проверка диапазонов для T2

if (ch2 < 0 || ch2 > 23 || min2 < 0 || min2 > 59 || sec2 < 0 || sec2 > 59) {

ShowMessage("Некорректные значения времени 2");

LabeledEdit2->SetFocus();

return;

}

// Создаем объекты через конструктор с параметрами

MyTime T1(ch1, min1, sec1);

MyTime T2(ch2, min2, sec2);

MyTime T3;

// Отображаем результаты

T1.show();

T2.show();

T3.summa(T1, T2);

T3.show();

}

catch (...) {

ShowMessage("Ошибка при обработке введенных данных");

}

}**Unit1.h**

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

#include <Vcl.ExtCtrls.hpp>

#include <Vcl.Mask.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TLabeledEdit \*LabeledEdit1;

TLabeledEdit \*LabeledEdit2;

TButton \*Button1;

TButton \*Button2;

TButton \*Button3;

TButton \*Button4;

TButton \*Button5;

TButton \*Button6;

TMemo \*Memo1;

TButton \*Button7;

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall Button2Click(TObject \*Sender);

void \_\_fastcall Button3Click(TObject \*Sender);

void \_\_fastcall Button4Click(TObject \*Sender);

void \_\_fastcall Button5Click(TObject \*Sender);

void \_\_fastcall Button6Click(TObject \*Sender);

void \_\_fastcall Button7Click(TObject \*Sender);

void \_\_fastcall LabeledEdit1Change(TObject \*Sender);

void \_\_fastcall LabeledEdit2Change(TObject \*Sender);

private: // User declarations

bool \_\_fastcall IsInputValid();

void \_\_fastcall UpdateButtonsState();

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**Unit1.h**

/ //---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

#include <Vcl.ExtCtrls.hpp>

#include <Vcl.Mask.hpp>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TMemo \*Memo1;

TButton \*Button1;

TLabeledEdit \*LabeledEdit1;

TLabeledEdit \*LabeledEdit2;

TButton \*Button2;

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall Button2Click(TObject \*Sender);

private: // User declarations

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

class MyTime {

private:

int chas;

int min;

int sec;

public:

MyTime() { chas=0; min=0; sec=0; }

MyTime(int ch, int m, int s) {

chas=ch; min=m; sec=s;

}

void show();

void summa(MyTime t1, MyTime t2);

bool parseTimeString(const AnsiString &timeStr);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**Вывод:** В ходе выполнения лабораторной работы по теме «Классы. Инкапсуляция» был успешно реализован класс MyTime для работы с временными значениями. Освоены базовые принципы объектно-ориентированного программирования: инкапсуляция данных в приватных полях класса и доступ к ним через методы. Реализованы два конструктора: по умолчанию и с параметрами, что позволило создавать объекты с разной начальной инициализацией. Созданы методы для отображения времени и сложения временных интервалов с корректной обработкой переполнения. Добавлен функционал пользовательского ввода временных значений с проверкой корректности формата. Реализованы два подхода к инициализации объектов: через метод parseTimeString и напрямую через конструктор с параметрами. Приобретенные навыки демонстрируют понимание принципов ООП в C++ и способность их применять в практических задачах.