Министерство науки и ВЫСШЕГО образования

Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Курский государственный университет»

Факультет физики, математики, информатики

Кафедра программного обеспечения и администрирования информационных систем

Курсовой проект

по дисциплине

Структуры и алгоритмы компьютерной обработки данных

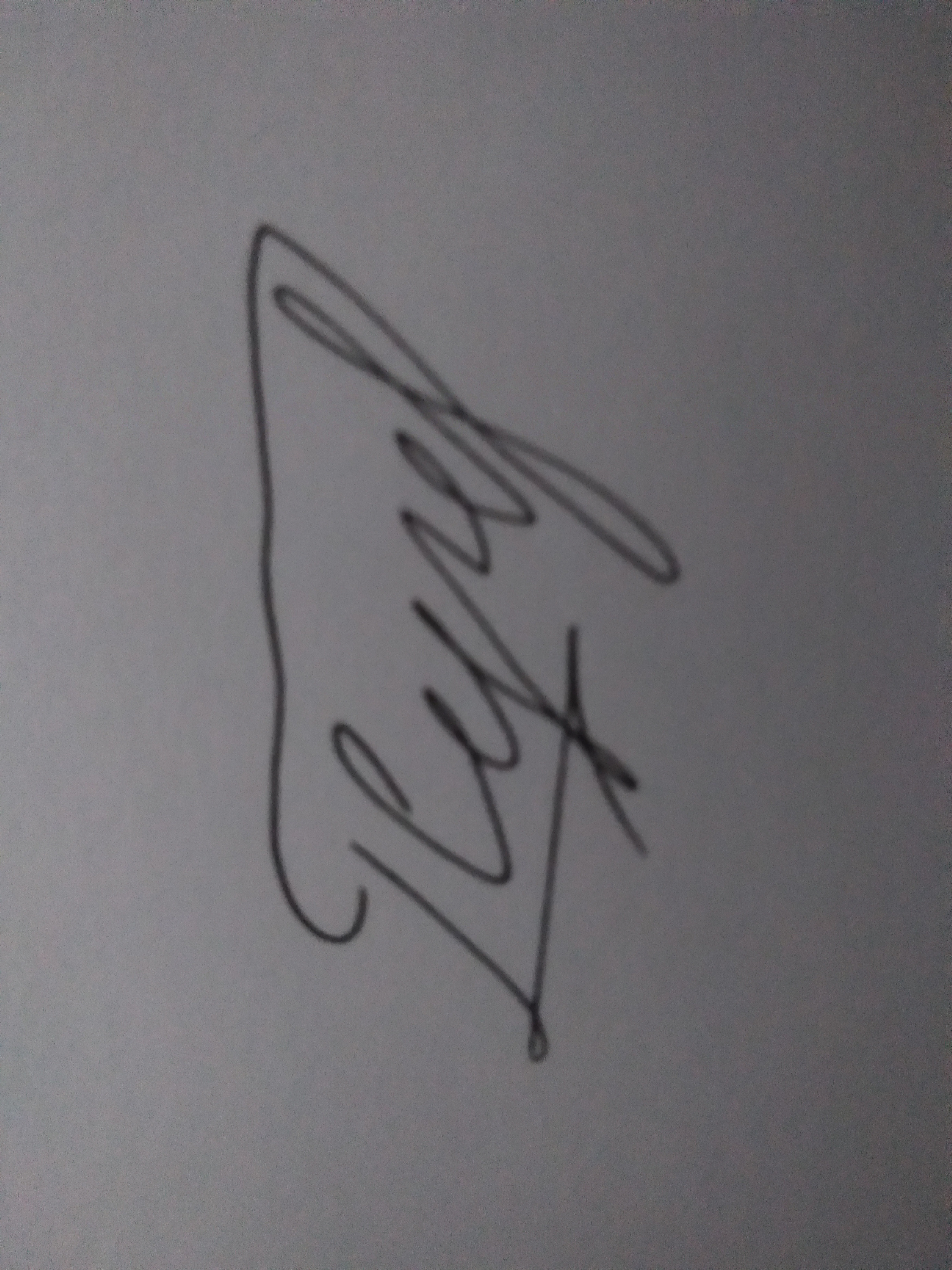
на тему: РАСПЕРЕДЕЛЁННЫЕ АЛГОРИТМЫ

Обучающегося 2 курса

очной формы обучения

направления подготовки

02.03.03 Математическое обеспечение и администрирование информационных систем

Направленность (профиль) Проектирование информационных систем и баз данных

Перетятько Сергея Игоревича

Руководитель:

профессор кафедры ПОиАИС

Кудинов Виталий Алексеевич

Допустить к защите:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«10» июня 2020 г.

Курск, 2020

АННОТАЦИЯ

В данной пояснительной записке содержатся описание и сведения по работе с приложением.

Ключевые слова и термины, используемые в приложении: мьютексы, взаимная блокировка, голодание, состояние гонки, процесс, база данных, чрезвычайно параллельные алгоритмы, многопроцессорные вычисления.

Число страниц: 45.

Число рисунков: 9.

Число приложений: 1.

Содержание

ВВЕДЕНИЕ4

1 Теоретический материал5

2 Структурное и функциональное описание приложения 10

Задача «Реализация мьютексов»11

«Задача византийских генералов»15

Задача «Синхронизация часов»20

3 Тестирование приложения22

4 Инструкция по работе с приложением26

5 Список используемых источников26

ПРИЛОЖЕНИЕ ТЕКСТ ПРОГРАММЫ27

ВВЕДЕНИЕ

Тема курсового проекта – распределённые алгоритмы, их разновидности и применение.

Было бы актуально создать иллюстрированную программу, которая понятно рассказывала бы пользователям об распределённых вычислениях, их важности и использовании в сфере компьютерных технологий. Это приложение могло бы стать хорошим справочником и наглядным пособием для студентов, обучающихся по направлениям технических специальностей, а также для тех, кто интересуется проблемами и задачами, решаемыми в сфере IT-индустрии.

Цель проекта – создать демонстрационную программу, в которой наиболее ясно и точно освещались бы проблемы и методы их решения в области распределённых вычислений и алгоритмов.

При реализации проекта необходимо решить ряд задач:

* создать удобный и понятный пользовательский интерфейс;
* кратко и понятно изложить теоретический материал для пользователей по исследуемой теме в виде справочной или энциклопедической информации;
* рассмотреть основные проблемы данной темы;
* программно реализовать несколько демонстрационных задач, в которых пользователи смогут сами вводить исходные данные, собрать и оттестировать рассматриваемые алгоритмы и получить соответствующие результаты по завершению их работы.

1 Теоретический материал

Существует несколько моделей параллелизма, и каждая из них основана на отдельном наборе допущений, в первую очередь количестве доступных процессоров и том, как они между собой соединены. Самой распространённой сегодня моделью являются распределённые вычисления.

Распределённые вычисления – это ситуация, когда несколько компьютеров или процессов работают вместе, например по сети, чтобы выполнить некую общую задачу. У каждого из них своя оперативная память, хотя жёсткие диски могут быть общими.

### Распределённые операционные системы

Распределённая ОС, динамически и автоматически распределяя работы по различным машинам системы для обработки, заставляет набор сетевых машин обрабатывать информацию параллельно. Пользователь распределённой ОС, вообще говоря, не имеет сведений о том, на какой машине выполняется его работа.

Распределённая ОС существует как единая операционная система в масштабах вычислительной системы. Каждый компьютер сети, работающей под управлением распределённой ОС, выполняет часть функций этой глобальной ОС. Распределённая ОС объединяет все компьютеры сети в том смысле, что они работают в тесной кооперации друг с другом для эффективного использования всех ресурсов компьютерной сети.

В результате сетевая ОС может рассматриваться как набор операционных систем отдельных компьютеров, составляющих сеть. На разных компьютерах сети могут выполняться одинаковые или разные ОС. Например, на всех компьютерах сети может работать одна и та же ОС [UNIX](https://ru.wikipedia.org/wiki/UNIX). Более реалистичным вариантом является сеть, в которой работают разные ОС, например, часть компьютеров работает под управлением UNIX, часть — под управлением NetWare, а остальные — под управлением [Windows NT](https://ru.wikipedia.org/wiki/Windows_NT) и [Windows 98](https://ru.wikipedia.org/wiki/Windows_98). Все эти операционные системы функционируют независимо друг от друга в том смысле, что каждая из них принимает независимые решения о создании и завершении своих собственных процессов и управлении локальными ресурсами. Но в любом случае операционные системы компьютеров, работающих в сети, должны включать взаимно согласованный набор коммуникационных протоколов для организации взаимодействия процессов, выполняющихся на разных компьютерах сети, и разделения ресурсов этих компьютеров между пользователями сети.

Если операционная система отдельного компьютера позволяет ему работать в сети, и может предоставлять свои ресурсы в общее пользование и/или использовать ресурсы других компьютеров сети, то такая операционная система отдельного компьютера также называется сетевой ОС.

Преимущества распределённых систем

- Обмен информацией. Необходимость обмена данными между различными компьютерами возросла в шестидесятых, когда большинство основных университетов и компаний начали пользоваться своими собственными майнфреймами. Взаимодействие между людьми из различных организаций облегчилось благодаря обмену данными между компьютерами этих организаций, и это дало рост развитию так называемых глобальных сетей (WAN). Компьютерная система соединенная в глобальную сеть обычно снабжалась всем что необходимо пользователю: резервными хранилищами данных, дисками, многими прикладными программами и принтерами. Позже компьютеры стали меньше и дешевле, и сегодня одна организация может иметь множество компьютеров, иногда даже один компьютер на одного работника (рабочую станцию). В этом случае также требуется чтобы эти компьютеры были соединены для электронного обмена информацией между персоналом компании.

- Разделение ресурсов. Хотя с приходом более дешевых компьютеров стало возможно снабжать каждого сотрудника организации личным компьютером, это же нельзя сделать для периферии (принтеры, резервные хранилища, блоки дисков). В этом меньшем масштабе каждый компьютер может положиться на специальные серверы, которые снабжают его компиляторами и другими прикладными программами. Также, памяти любого компьютера обычно недостаточно, чтобы хранить большой набор прикладных программ, требуемых для каждого пользователя. Кроме того, компьютеры могут использовать специальные узлы для служб печати и хранения данных. Сеть, соединяющая компьютеры в масштабе предприятия называется локальной вычислительной сетью(LAN). Причины, по которым организация устанавливает сеть небольших компьютеров, а не майнфреймы – снижение стоимости и расширяемость. Во-первых, меньшие компьютеры имеют лучше соотношение цена-производительность, чем большие компьютеры. Типичный майнфрейм может совершать операции в 50 раз быстрее, чем персональный компьютер, но иметь стоимость в 500 раз большую. Во-вторых, если  мощности системы больше не достаточно, то сеть может быть расширена добавлением других машин (файловых серверов, принтеров и рабочих станций). Если мощность монолитной системы больше неудовлетворительна, остается только полная замена.

- Большая надежность благодаря репликации. Распределенные системы имеют потенциал надежности больший, чем монолитные системы благодаря свойству их частичного выхода из строя. Это значит, что некоторые узлы системы могут выйти из строя, в то время как другие по прежнему функционируют и могут взять на себя задачи испорченных компонентов. Выход из строя монолитного компьютера действует на всю систему целиком и нет возможности продолжать вычисления в этом случае. По этой причине распределенные архитектуры представляют интерес при разработке высоко надежных компьютерных систем. Высоко надежная система обычно состоит из двух, трех или четырех репликационных унипроцессоров, которые исполняют прикладную программу и поддерживаются механизмом голосования, чтобы отфильтровывать результаты машин. Правильное функционирование  распределенной системы при наличии поврежденных компонент требует довольно сложной алгоритмической поддержки.

- Большая производительность благодаря распараллеливанию. Наличие многих процессоров в распределенной системе открывает возможность снижения дополнительного времени для интенсивной работы с помощью разделения работы среди нескольких процессоров. Параллельные компьютеры разработаны специально для этой цели, но пользователи локальных сетей также могут получить пользу от параллелизма, перекладывая задачи на другие рабочие станции.

- Упрощение разработки благодаря специализации. Разработка компьютерной системы может быть сложной, особенно если требуется значительная функциональность. Разработка может быть зачастую упрощена разбитием системы на модули, каждый из которых отвечает за часть функциональности и коммутируется с другими модулями. На уровне одной программы модульность достигается определением абстрактных типов данных и процедур для различных задач. Большая система может быть определена как набор кооперирующих процессов. В обоих случаях, модули могут быть исполнены в рамках одного компьютера. Но также возможно иметь локальную сеть с различными типами компьютеров: один снабжен специальным оборудованием для вычислений, другой – графическим оборудованием, третий – дисками и т.д.

Чрезвычайно параллельные алгоритмы

Чрезвычайно параллельными называют алгоритмы, которые естественным образом разбиваются на самостоятельные задачи, хорошо подходящие для решения отдельными процессами. Это не требует интенсивного взаимодействия между процессами; объединение результатов в идеале должно выполняться довольно легко.

Примеры чрезвычайно параллельных задач.

- В компьютерной графике применяется методика, в которой луч трассируется из точки обзора на сцену, чтобы увидеть, какие объекты он задевает.

- Обслуживание статических файлов на [веб-сервере](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80).

- Расчёт элементов [множества Мандельброта](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%BE_%D0%9C%D0%B0%D0%BD%D0%B4%D0%B5%D0%BB%D1%8C%D0%B1%D1%80%D0%BE%D1%82%D0%B0) и других [фракталов](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B0%D0%BA%D1%82%D0%B0%D0%BB), когда каждая точка может быть вычислена независимо.

- Поиск методом полного перебора.

- Поиск по неиндексированной базе данных.

- [Генетические алгоритмы](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) и другие [эвристические алгоритмы](https://ru.wikipedia.org/wiki/%D0%AD%D0%B2%D1%80%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC).

- Моделирование и реконструкция событий в [физике элементарных частиц](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D0%BA%D0%B0_%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%BD%D1%8B%D1%85_%D1%87%D0%B0%D1%81%D1%82%D0%B8%D1%86).

- Этап сбора отношений в вариации [кратных](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5) [многочленов](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD) метода квадратичного решета — [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) в [факторизации](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [целых чисел](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D1%8B%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%B0) (MPQS).

Проблемы, возникающие при реализации распределённых алгоритмов.

- Состояние гонки. Состояние гонки означает, что два процесса пытаются выполнить запись внутрь ресурса примерно в один и тот же момент времени. Процесс, который записывает вторым, побеждает.

- Перестановка приоритетов. Высокоприоритетный процесс блокируется в ожидании низкоприоритетного, владеющего мьютексом.

- Голодание. Процесс не может получить ресурсы, которые ему нужны для завершения работы. Иногда это происходит в ситуациях, когда операционная система пытается решить проблему перестановки приоритетов. Если высокоприоритетная задача загружает центральный процессор, у процесса с низким приоритетом может иногда не появиться шанса быть выполненным, из-за чего он завершится.

- Взаимная блокировка. Два процесса блокируются в ожидании друг друга.

2 Структурное и функциональное описание приложения

Приложение представляет собой объединение 3 демонстрационных задач: «реализация мьютексов», «задача византийских генералов», «синхронизация часов», и 6 окон, составляющих справочную информацию, содержащуюся в нём и предоставляемую пользователю.

Программа состоит из нескольких модулей, форм.

Функционирование программ приложения формируется на базе следующих модулей:

- Unit1.cpp – модуль, который отвечает за взаимодействие окна входа с пользователем.

- Unit2.cpp – модуль, в котором находится код работы главного меню приложения.

- Unit3.cpp – программа, с помощью которой реализована демонстрационная задача на реализацию мьютексов.

- Unit4.cpp – модуль, обеспечивающий возможность пользователю создать ситуации работы процессов, которые могут произойти при использовании мьютексов.

- Unit5.cpp – файл модуля, который отвечает за реализацию задачи византийских генералов.

- Unit6.cpp – модуль программы, в котором содержится алгоритм на синхронизацию часов.

- Unit7.cpp, Unit8.cpp, Unit9.cpp, Unit10.cpp, Unit11.cpp, Unit12.cpp – модули, составляющие справочную систему программного продукта.

Задача «Реализация мьютексов»

Мьютекс ([англ.](https://ru.bmstu.wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *mutex*, от [англ.](https://ru.bmstu.wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *mutual exclusion* — «взаимное исключение») в программировании необходим для сопоставления синхронно выполняющихся потоков. Мьютекс представляет собой концепцию программирования, которая используется для решения вопросов многопоточности. Мьютекс допускает только один поток в контролируемом участке, заставляя другие потоки, которые пытаются получить доступ к этому разделу ждать, пока первый поток не вышел из этого раздела.

Принимает два значенения:

открыт (разблокирован)- поток может войти в свою критическую секцию;

закрыт (заблокирован) - поток не может войти в критическую секцию.

Задача мьютекса — защита объекта от доступа к нему других потоков, отличных от того, который завладел мьютексом. В каждый конкретный момент только один поток может владеть объектом, защищённым мьютексом. Если другому потоку будет нужен доступ к переменной, защищённой мьютексом, то этот поток засыпает до тех пор, пока мьютекс не будет освобождён.

Цель использования мьютексов — защита данных от повреждения в результате асинхронных изменений (состояние гонки), однако могут порождаться другие проблемы — такие, как взаимная блокировка (клинч). Для описания мьютекса требуется всего один бит, хотя чаще используется целая переменная Значение мьютекса устанавливается двумя процедурами. Если мьютекс не заблокирован, запрос выполняется и вызывающий поток может попасть в критическую область. Если mutex закрыт, то поток пытающийся войти в критическую секцию блокируется.

Алгоритм реализации решения для данной задачи содержится в программных модулях Unit3.cpp и Unit4.cpp. Unit4.cpp даёт возможность пользователю самому синтезировать на выбор 3 процесса на обработку общего фрагмента памяти из предоставленного ему набора команд. После чего можно нажать кнопку запуска и посмотреть работу алгоритмов построенных процессов.

Ниже представлены структуры данных и функции программы, с помощью которых была реализована задача.

AnsiString Commarr[15] – массив, содержащий набор команд, предоставляемых пользователю для построения процессов.

int Procarr[3][10] – данная структура хранит в себе 3 процесса, которые состоят из набора команд, выбранных пользователем для их построения. Каждая строка массива символизирует отдельный процесс, каждый столбец – команды, которые будут выполнять процессы в один и тот же такт дискретного времени.

int i1, i2, i3 – переменные счётчики, которые определяют выполняемые команды соответствующих процессов в данный такт времени.

bool l1, l2, l3 – логические переменные, которые определяют состояние процесса: он может выполняться или быть заблокированным в ожидании освобождения нужного ему мьютекса.

int a, b, c. Каждая из этих переменных связана с соответствующей ей ячейкой памяти. Она определяет, каким процессом захвачена ячейка. Если значение равно 0, то это означает, что данная ячейка ни одному из существующих процессов не принадлежит.

void showcom (int, int, TEdit \*) – функция, которая выводит на экран команды всех 3 процессов, выполняющихся в данный такт дискретного времени.

void f1 (TEdit \*, TLabel \*) – подпрограмма, реализующая команду ввода данных пользователем, которые передадутся во временную переменную для определённого процесса.

void f2 (TStringGrid \*, TLabel \*, bool &, int &, int i) – функция, ассоциирующаяся с командой определённого процесса на вывод данных в ячейку A. Если ячейка принадлежит какому-либо другому процессу, то запись не происходит, а данный процесс переходит в заблокированное состояние.

void f3 (TStringGrid \*, TLabel \*, bool &, int &, int) - функция, реализующая команду вывода значения в ячейку B процессом. Если ячейка принадлежит какому-либо другому процессу, то значение не записывается, а данный процесс блокируется.

void f4 (TStringGrid \*, TLabel \*, bool &, int &, int ) - функция, реализующая команду вывода значения в ячейку C процессом. Запись данных происходит в том случае, если ячейка используется другим процессом, в противном случае процесс, выполняющий данную команду переходит в заблокированное состояние.

void f5\_6\_7 (int &, int, bool &) – подпрограмма операции процесса на захват ячейки. Информация о, том, какую именно ячейку памяти следует взять под контроль, передаётся в параметре функции. Если переменная уже захвачена другим процессом, данного захвата мьютекса не происходит, а соответствующий алгоритм блокируется.

void f8 (TLabel \*, TEdit \*) - подпрограмма, реализующая команду вывода данных пользователю, которые хранятся во временной переменной определённого процесса. Номер процесса указывается в параметре функции.

void f9\_10\_11 (int &, int, bool &) – функция, отвечающая за команду освобождения заданной ячейки. В параметре функции указывается, какую ячейку следует освободить процессу. После освобождения остальные процессы, которые были на очереди захвата этой ячейки памяти, переходят из заблокированного состояния в состояние выполнения.

void f12 ( TLabel \*, TStringGrid \*, bool &, int &, int) – подпрограмма, которая выполняет команду процесса на считывание данных из ячейки A. Данные могут считаться только в том случае, если ячейка не занята остальными процессами, иначе считывания не происходит и процесс, указанный в параметре функции, блокируется.

void f13 ( TLabel \*, TStringGrid \*, bool &, int &, int) – процедура по реализации считывания значения из ячейки B процессом, указанным в параметре. Если переменная, из которой необходимо считать значение занята другим процессом, то данный алгоритм переходит в заблокированное состояние.

void f14 ( TLabel \*, TStringGrid \*, bool &, int &, int) – процедура, которая обеспечивает считывание значения из ячейки C. Если ячейка уже занята, то значение не передаётся процессу во временную переменную и он блокируется в ожидании освобождения выбранной ячейки памяти.

Интерфейс и управляющие компоненты формы для задачи на реализацию мьютексов

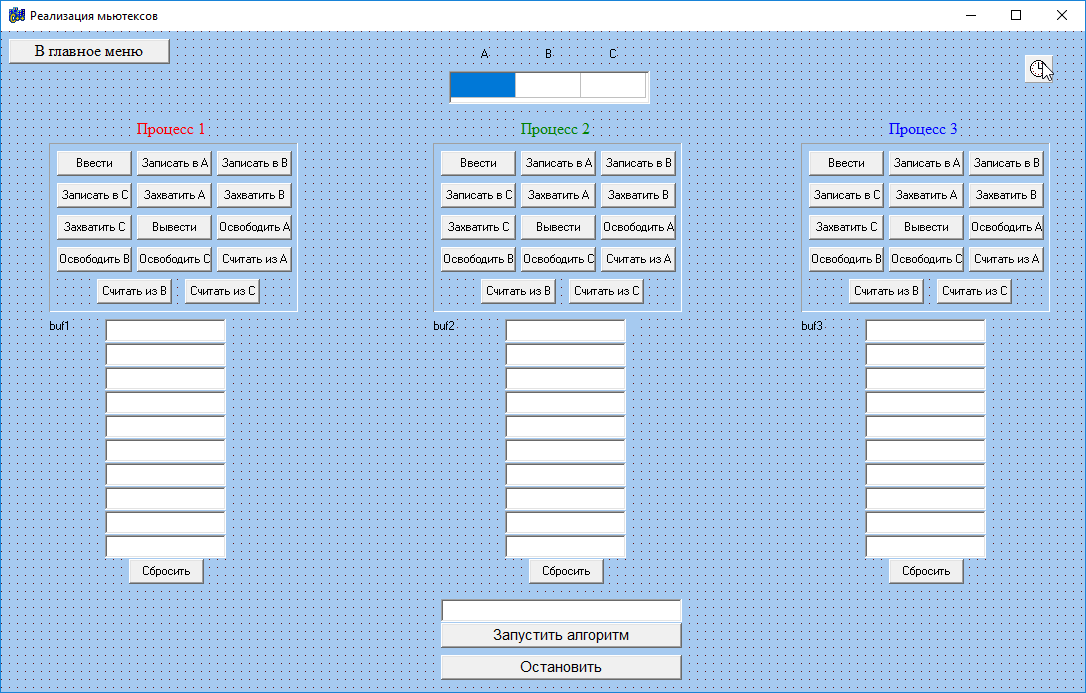


Рисунок 1 – Форма для построения алгоритма работы с мьютексами

Набор кнопок A1-A14 задаёт все команды, которые могут быть посланы пользователем на выполнение процессом 1.

Кнопки B1-B14 задают набор команд для процесса 2.

Кнопки C1-C14 – все возможные команды для процесса 3.

Edit – элемент, дающий возможность пользователю ввести значение, которое затем может быть передано одному из процессов на обработку.

Button1 – кнопка, которая очищает алгоритм процесса 1.

Button2 – кнопка сброса алгоритма процесса 2.

Button3 – кнопка очищения алгоритма процесса 3.

Button4 – главная кнопка запуска работы процессов.

Button5 – кнопка остановки выполнения процессов.

FlashData – массив, реализующий фрагмент памяти, который предназначен для обработки процессами.

Timer1 – таймер, который задаёт такт времени на выполнение одной команды. Такт равен 2 секундам. Сначала таймер не активирован. Таймер работает, только если запущена программа на выполнение процессов.

«Задача византийских генералов»

Задача византийских генералов ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Byzantine fault tolerance (BFT), Byzantine agreement problem, Byzantine generals problem, Byzantine failure) — в [криптологии](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) задача взаимодействия нескольких удалённых абонентов, которые получили приказы из одного центра. Часть абонентов, включая центр, могут быть злоумышленниками. Нужно выработать единую стратегию действий, которая будет выигрышной для абонентов.

[Византия](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D0%B0%D0%BD%D1%82%D0%B8%D1%8F). Ночь перед великим сражением с противником. Византийская армия состоит из легионов, каждым из которых командует свой генерал. Также у армии есть главнокомандующий, которому подчиняются генералы.

В то же самое время, империя находится в упадке, и любой из генералов и даже главнокомандующий могут быть предателями Византии, заинтересованными в её поражении.

Ночью каждый из генералов получает от предводителя приказ о варианте действий в 10 часов утра (время одинаковое для всех и известно заранее), а именно: «атаковать противника» или «отступать».

Возможные исходы сражения:

Если все верные генералы атакуют — Византия уничтожит противника (благоприятный исход).

Если все верные генералы отступят — Византия сохранит свою армию (промежуточный исход).

Если некоторые верные генералы атакуют, а некоторые отступят — противник уничтожит всю армию Византии (неблагоприятный исход).

Также следует учитывать, что если главнокомандующий — предатель, то он может дать разным генералам противоположные приказы, чтобы обеспечить уничтожение армии. Следовательно, генералам лучше не доверять его приказам.

Если же каждый генерал будет действовать полностью независимо от других (например, сделает случайный выбор), то вероятность благоприятного исхода весьма низка.

Поэтому генералы нуждаются в обмене информацией между собой, чтобы прийти к единому решению.

Сначала главнокомандующий раздаёт приказы всем генералам на атаку или отступление. После этого все генералы между собой обмениваются отданными им приказами. После чего каждый генерал имеет сведения о всех приказах, которые главнокомандующий отдал всем генералам. После чего происходит ещё один обмен информацией, после которого генералы имеют составленную таблицу всех сообщённых приказов от всех лиц. Если в таблице все приказы одинаковые, то это значит, что все генералы верные, в том числе и главнокомандующий войсками. Если в столбцах приказы одинаковые, а в строках имеются различия, то это означает, что предателем является главнокомандующий, а все подчиняющиеся генералы верные. Они могут большинством голосов принять приказ о общем наступлении или отступлении. Если в одном столбце есть различия, а во всех остальных содержится одинаковый приказ, это значит, что предатель – тот генерал, у которого в столбце приказы оказались разными. В этом случае большинством голосов будет принято решение о наступлении или отступлении. Если различия есть и в строках и в столбцах, то генералы не смогут договориться и армия потерпит поражение в сражении.

Программа к решению задачи написана в модуле Unit5.cpp

Ниже представлены структуры данных и функции программы, с помощью которых была реализована задача.

class General – класс, иллюстрирующий генерала.

Его поля и методы:

AnsiString action – строка, содержащая приказ от главнокомандующего.

int num – задаёт номер данного генерала, с которым будут обмениваться сведениями другие.

bool D – логическая переменная, хранящая информацию про то, какой приказ был передан. 0 – приказ на отступление, 1 – на атаку.

AnsiString Allpr[4] – массив, содержащий все приказы после обмена данными с другими генералами.

General(int ) – конструктор класса по созданию объекта генерала. В качестве параметра передаётся его номер.

void Poluchpr(AnsiString ) – метод получения приказа от главнокомандующего.

void tallpr(General &, AnsiString) – метод, который позволяет сообщить генералу другим о том, какой приказ был ему передан.

void showpr(TStringGrid \*T) – функция класса на вывод сведений о всех приказах, которые сообщили данному другие генералы.

class Commander: public General – класс, реализующий главнокомандующего. Является потомком класса General.

Его поля и методы:

bool ListGetPr[4], AnsiString ListPr[4]; – массивы, хранящие приказы, которые были переданы главнокомандующим генералам напрямую.

void SendPrikaz (AnsiString, int, General &) – метод, который отвечает за передачу приказа от главнокомандующего к одному из генераов. В параметре метода указывается номер генерала, которому будет передан данный приказ.

Commander(int) – конструктор по созданию объекта класса «главнокомандующий».

bool Sved[4][4] – массив со всеми сообщениями о приказах, полученных всеми генералами после обмена.

int Zased[4] – массив, с помощью которого выявляется информация о предателе и определяется общее решение на атаку всех войск или отступление.

int Polog – переменная, определяющая ситуацию. Если значение равно 4, то это означает, что предателей несколько, генералы не могут договориться, в результате чего армия либо проиграет, либо взбунтуется. 3 – среди генералов есть один предатель, который может быть выявлен. На основании большинства голосов будет принято общее решение. 2 означает, что предателем является главнокомандующий войсками, но верные генералы всё равно смогут договориться. Значения 1 и 0 сообщают, что все генералы, включая главнокомандующего, верны. 0 – решение на отступление, 1 – приказ на атаку.

Интерфейс и управляющие компоненты формы для решения задачи византийских генералов

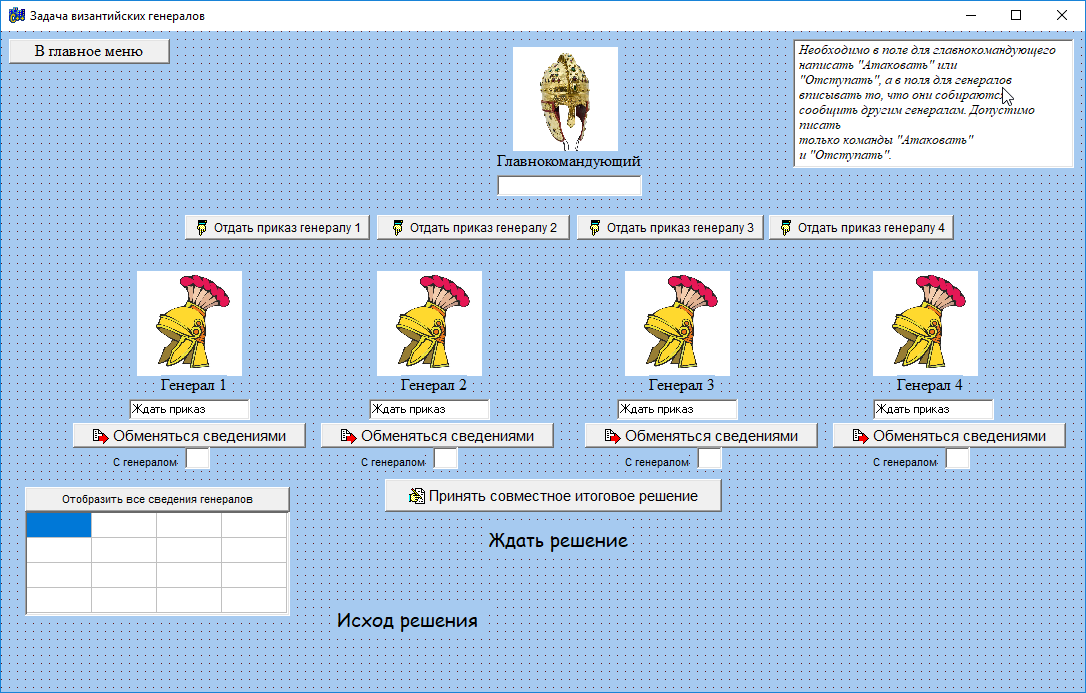


Рисунок 2 – Форма для проектирования решения задачи византийских генералов

BitBtn1 – кнопка для принятия итогового совместного решения всеми генералами.

BitBtn2 – кнопка передачи приказа от главнокомандующего генералу 1.

BitBtn3 – кнопка передачи приказа от главнокомандующего генералу 2.

BitBtn4 - кнопка передачи приказа от главнокомандующего генералу 3.

BitBtn5 - кнопка передачи приказа от главнокомандующего генералу 4.

Edit1, Edit2, Edit3, Edit4 – поля, в которых содержится информация генералов, которая будет сообщена другим (она может истинной, а может быть и ложной).

Edit5, Edit6, Edit7, Edit8 – поля, в которых содержатся номера генералов, которым будет передаваться информация.

PrTab – таблица, в которой будут отображаться все сведения всех генералов, после их обмена сообщениями.

Button1 – кнопка, которая отображает всю информацию в PrTab.

Memo1 – информация для пользователя о том, как нужно вводить информацию.

В метках выводится информация по принятию итогового совместного решения.

Задача «Синхронизация часов»

Ниже представлены структуры данных, с помощью которых была реализована задача.

int E – переменная, определяющая отклонение во времени между двумя часами.

int D – значение, определяющее задержку, необходимую для передачи сообщения от одного процесса к другому.

int TimClock1[3] – массив, хранящий информацию о текущем времени, которое отображается в первых часах.

int TimClock2[3] - массив, который хранит текущее время в часах и минутах, которое отображается во вторых часах.

int Ta1, int Ta2, int Tb1, int Tb2 – переменные, которые означают сообщения, передаваемые от одним часам к другим. В них содержится время одних из часов, переведённое в количество минут.

Edit5 – поле для ввода задержки, необходимой для передачи сообщения от одних часов к другим.

Интерфейс и управляющие компоненты формы для решения задачи на синхронизацию часов

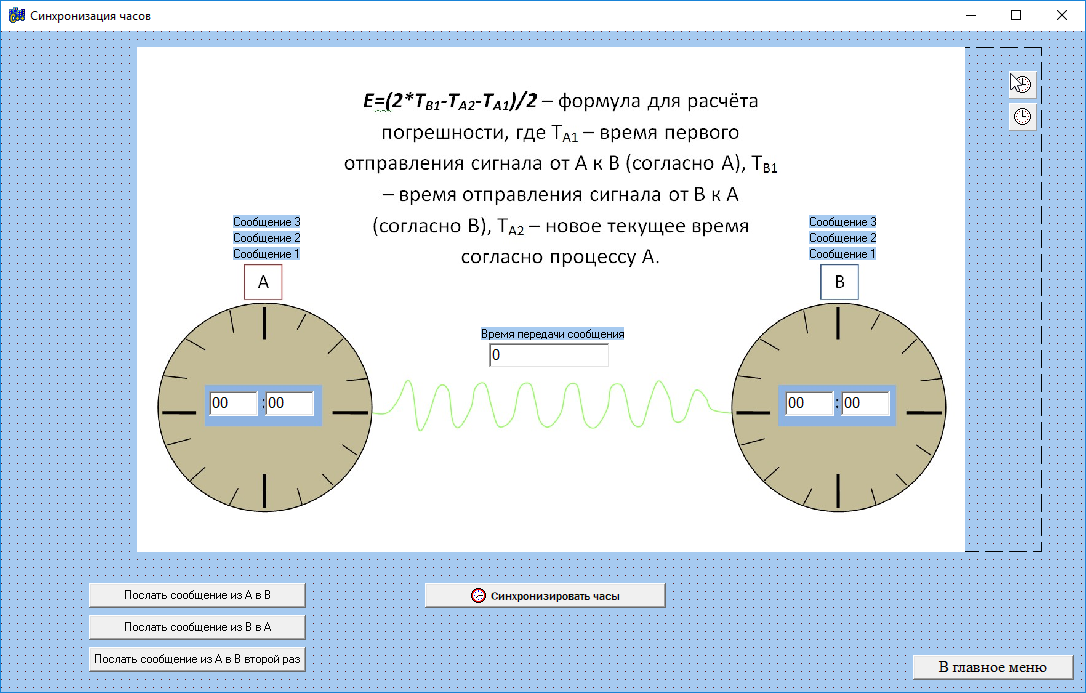


Рисунок 3 – Внешний вид окна, на котором реализован процесс синхронизации часов

BitBtn1 – кнопка, при нажатии на которую вторые часы синхронизируются с первыми.

Timer1 – таймер, который каждую секунду отсчитывает с помощью счётчика показания времени на часах.

Timer2 – таймер, который нужен для выполнения стадий процесса по синхронизации часов.

В модуле проекта Unit7.cpp представлена информация о распределённых алгоритмах и истории их появления. В Unit8.cpp содержатся сведения о применении распределённых алгоритмов. В Unit9.cpp описаны их свойства и виды.

В модуле Unit10.cpp описана проблема состояния гонки, в Unit11.cpp – проблема голодания. Unit12.cpp содержит информацию о состоянии взаимной блокировки.

3 Тестирование приложения

Ниже представлены рисунки, на которых отображена работа программных модулей приложения.

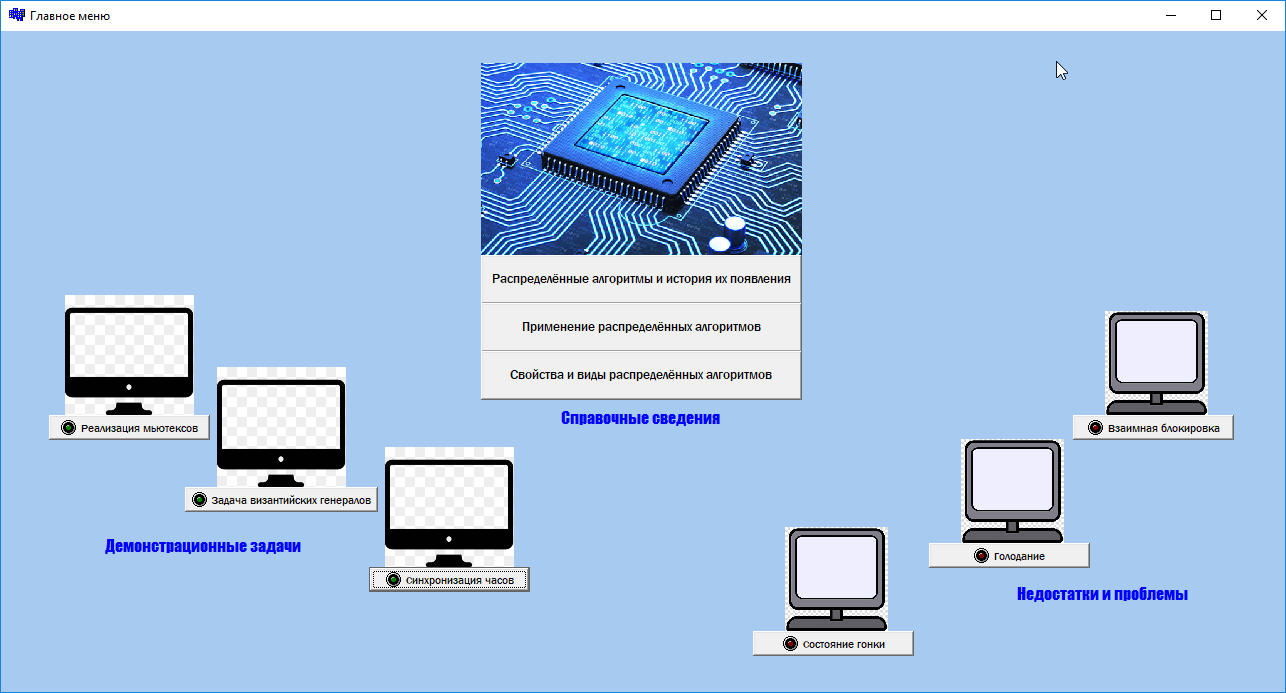


Рисунок 4 – Работа главного меню

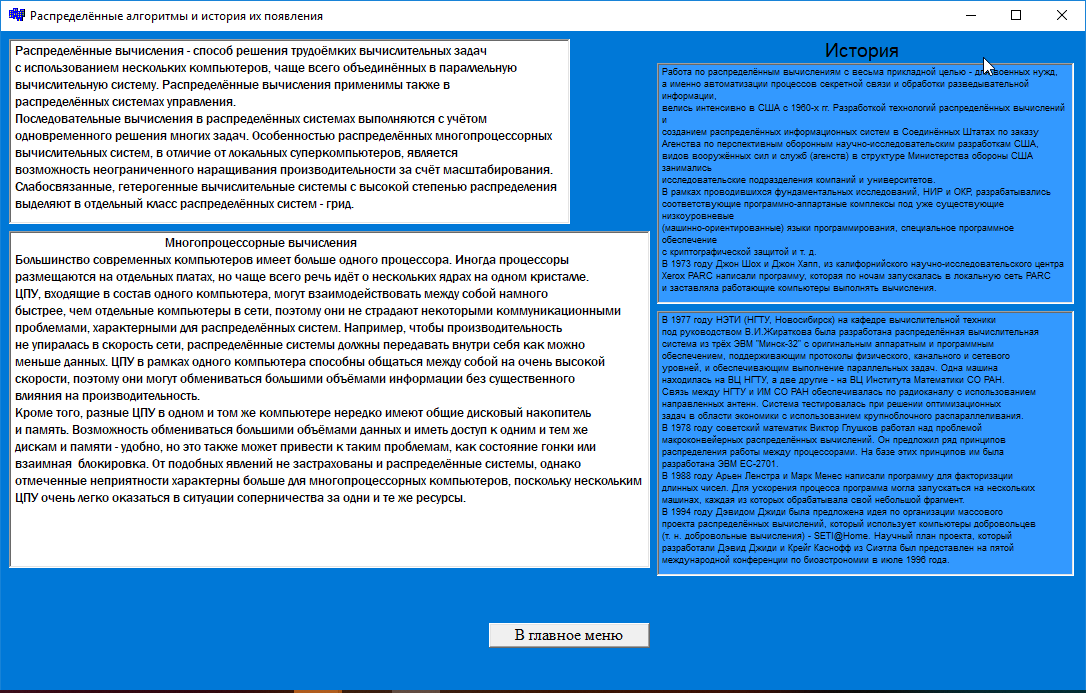


Рисунок 5 – Окно со справочной информацией по распределённым алгоритмам

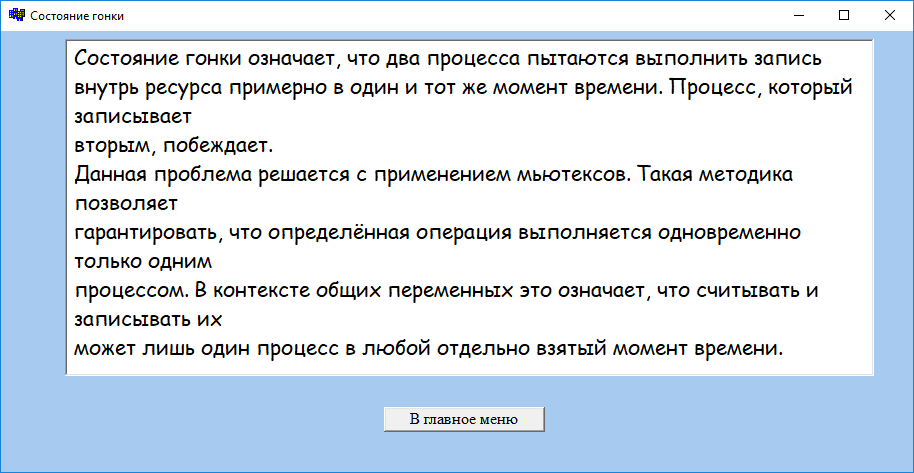


Рисунок 6 – Окно с описанием проблемы состояния гонки

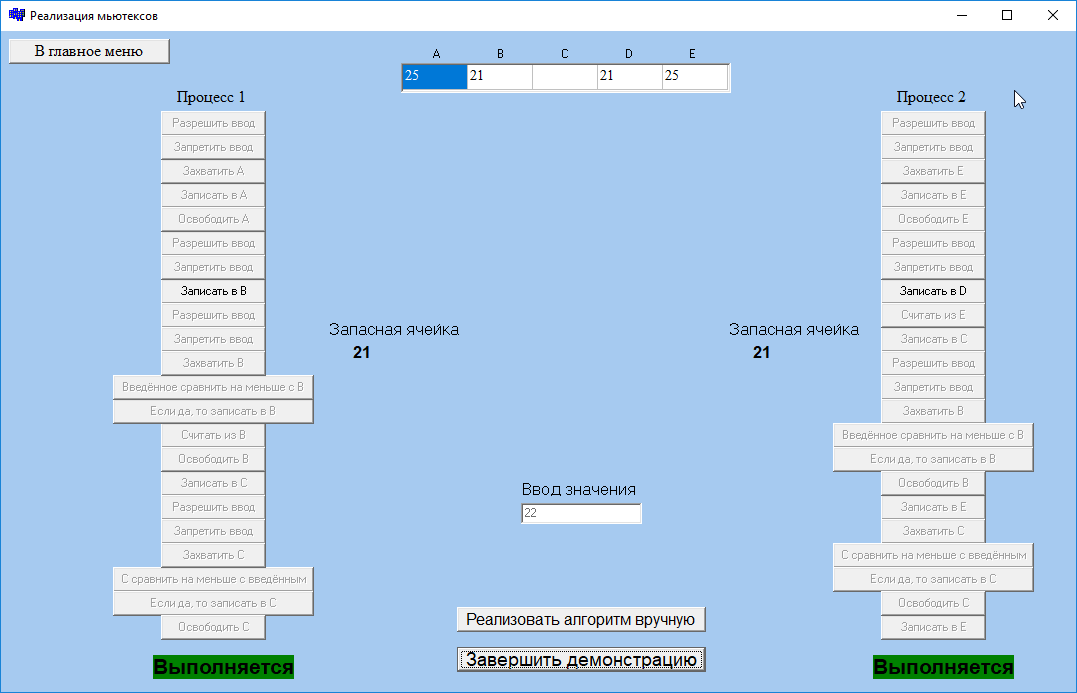


Рисунок 7 – Выполнение демонстрационной программы на реализацию работы с мьютексами

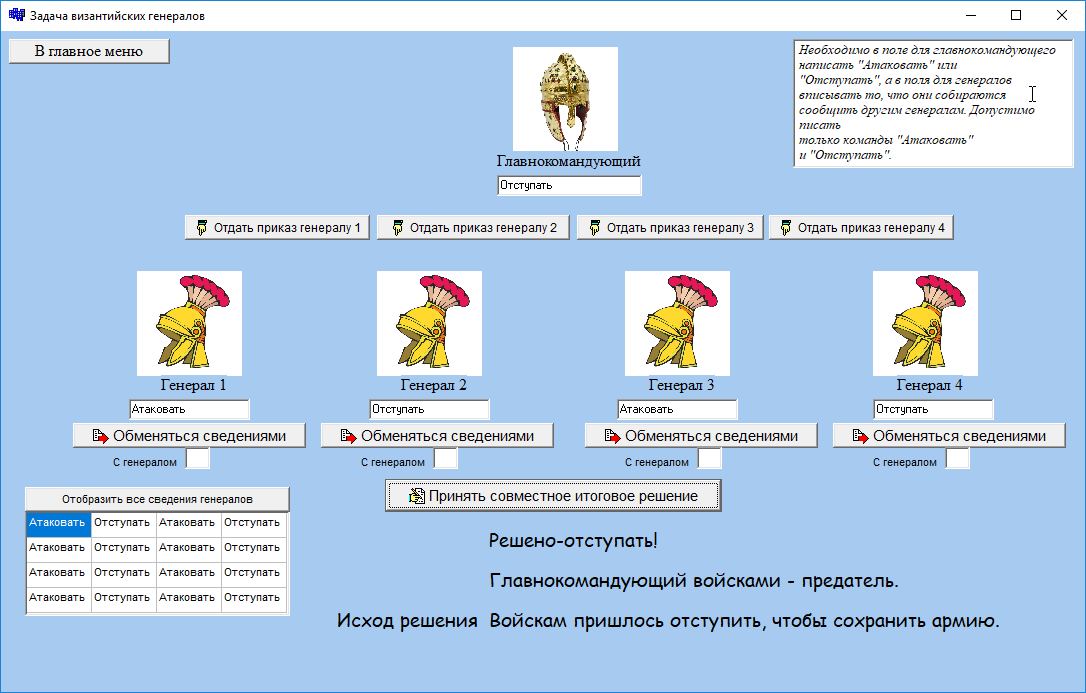


Рисунок 8 – Тестирование решения задачи византийских генералов

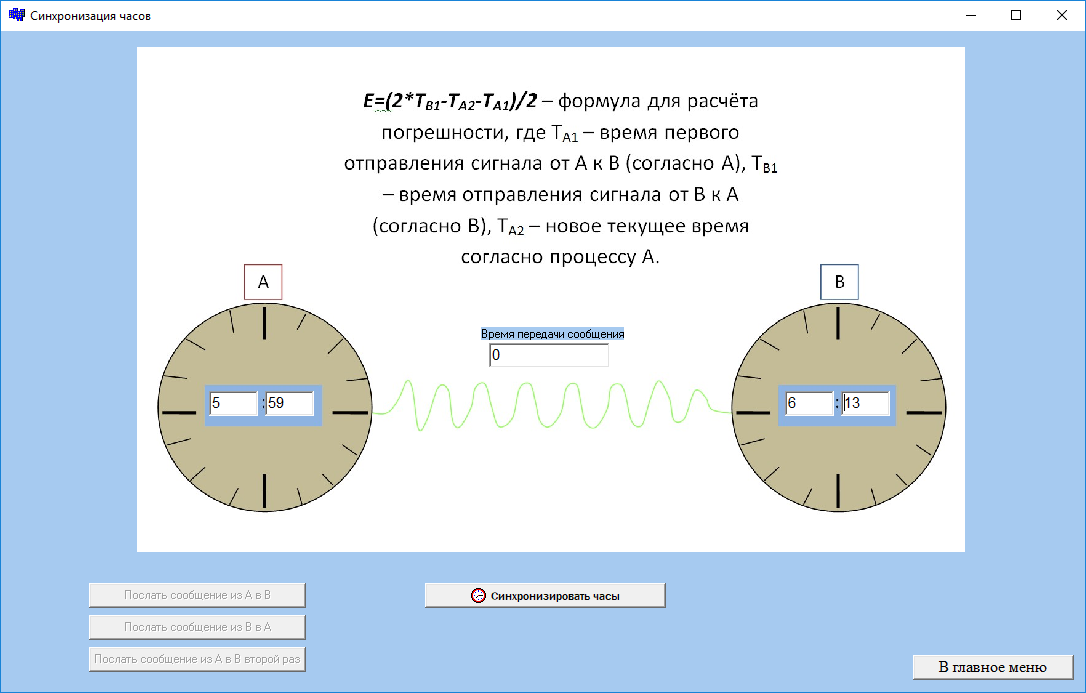


Рисунок 9 – Демонстрация синхронизации часов

4 Инструкция по работе с приложением

Для работы с приложением необходимо запустить файл KURSOVAYA\_SAKOD.exe. После чего пользователь войдёт в начальное окно входа, на котором нужно будет нажать кнопку войти. После этого откроется главное меню с различными рубриками.

5 Список используемых источников

1. Алгоритмы. Теория и практическое применение / Род Стивенс. – Москва: Издательство «Э», 2017. – 544 с.
2. Шилдт Г. С++ для начинающих.– СПб.: ЭКОМ, 2013. – 640 с.
3. Архангельский А.Я., Тагин М.А. Программирование в С++ Builder 6 и 2006. – М.: ООО «Бином-Пресс», 2007. – 1184 с.

ПРИЛОЖЕНИЕ ТЕКСТ ПРОГРАММЫ

Текст модуля Unit1.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TFirstForm \*FirstForm;

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

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

: TForm(Owner)

{

}

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

void \_\_fastcall TFirstForm::BitBtn1Click(TObject \*Sender)

{

Mainmenu->Visible=true;

Mainmenu->Show();

FirstForm->Visible=false;

}

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

Текст модуля Unit2.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit2.h"

#include "Unit3.h"

#include "Unit5.h"

#include "Unit6.h"

#include "Unit7.h"

#include "Unit8.h"

#include "Unit9.h"

#include "Unit10.h"

#include "Unit11.h"

#include "Unit12.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TMainmenu \*Mainmenu;

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

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

: TForm(Owner)

{

}

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

void \_\_fastcall TMainmenu::BitBtn2Click(TObject \*Sender)

{

ByzantGenerals->Visible=true;

ByzantGenerals->Show();

Mainmenu->Close();

}

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

void \_\_fastcall TMainmenu::BitBtn1Click(TObject \*Sender)

{

Realmutex->Visible=true;

Realmutex->Show();

Mainmenu->Close();

}

void \_\_fastcall TMainmenu::BitBtn3Click(TObject \*Sender)

{

ClockSyn->Visible=true;

ClockSyn->Enabled=true;

ClockSyn->Show();

Mainmenu->Close();

}

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

void \_\_fastcall TMainmenu::BitBtn4Click(TObject \*Sender)

{

Form10->Visible=true;

Form10->Show();

Mainmenu->Close();

}

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

void \_\_fastcall TMainmenu::BitBtn5Click(TObject \*Sender)

{

Form11->Visible=true;

Form11->Show();

Mainmenu->Close();

}

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

void \_\_fastcall TMainmenu::BitBtn6Click(TObject \*Sender)

{

Form12->Visible=true;

Form12->Show();

Mainmenu->Close();

}

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

void \_\_fastcall TMainmenu::Button4Click(TObject \*Sender)

{

ByzantGenerals->Visible=true;

ByzantGenerals->Show();

Mainmenu->Close();

}

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

void \_\_fastcall TMainmenu::B1Click(TObject \*Sender)

{

Form7->Visible=true;

Form7->Enabled=true;

Form7->Show();

Mainmenu->Close();

}

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

void \_\_fastcall TMainmenu::B2Click(TObject \*Sender)

{

Form8->Visible=true;

Form8->Enabled=true;

Form8->Show();

Mainmenu->Close();

}

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

void \_\_fastcall TMainmenu::B3Click(TObject \*Sender)

{

Form9->Visible=true;

Form9->Enabled=true;

Form9->Show();

Mainmenu->Close();

}

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

Текст модуля Unit3.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit3.h"

#include "Unit4.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TRealmutex \*Realmutex;

unsigned short int k=0;

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

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

: TForm(Owner)

{

}

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

bool Offon = false;

void \_\_fastcall TRealmutex::Timer1Timer(TObject \*Sender)

{

k++;

bool f=false;

if (k==1)

{

Label1->Caption="Выполняется";

Label1->Color=clGreen;

Label2->Caption="Выполняется";

Label2->Color=clGreen;

A1->Enabled=true;

B1->Enabled=true;

Edit1->Enabled=true;

Edit1->Text="25";

}

if (k==2)

{

A2->Enabled=true;

B2->Enabled=true;

A1->Enabled=false;

B1->Enabled=false;

Edit1->Enabled=false;

Buf1->Caption="25";

Buf2->Caption="25";

}

if (k==3)

{

A3->Enabled=true;

B3->Enabled=true;

A2->Enabled=false;

B2->Enabled=false;

sv1->Caption="Захвачен пр.1";

sv5->Caption="Захвачен пр.2";

}

if (k==4)

{

A4->Enabled=true;

B4->Enabled=true;

A3->Enabled=false;

B3->Enabled=false;

FlashData->Cells[0][0]=Buf1->Caption;

FlashData->Cells[4][0]=Buf2->Caption;

}

if (k==5)

{

A5->Enabled=true;

B5->Enabled=true;

A4->Enabled=false;

B4->Enabled=false;

sv1->Caption="";

sv5->Caption="";

}

if (k==6)

{

A6->Enabled=true;

B6->Enabled=true;

A5->Enabled=false;

B5->Enabled=false;

Edit1->Enabled=true;

Edit1->Text="22";

}

if (k==7)

{

A7->Enabled=true;

B7->Enabled=true;

A6->Enabled=false;

B6->Enabled=false;

Edit1->Enabled=false;

Buf1->Caption="21";

Buf2->Caption="21";

}

if (k==8)

{

A8->Enabled=true;

B8->Enabled=true;

A7->Enabled=false;

B7->Enabled=false;

FlashData->Cells[1][0]=Buf1->Caption;

FlashData->Cells[3][0]=Buf2->Caption;

}

if (k==9)

{

A9->Enabled=true;

B9->Enabled=true;

A8->Enabled=false;

B8->Enabled=false;

Edit1->Enabled=true;

Edit1->Text="60";

Buf2->Caption=FlashData->Cells[4][0];

}

if (k==10)

{

A10->Enabled=true;

B10->Enabled=true;

A9->Enabled=false;

B9->Enabled=false;

FlashData->Cells[2][0]=Buf2->Caption;

Buf1->Caption=Edit1->Text;

Edit1->Enabled=false;

}

if (k==11)

{

A11->Enabled=true;

B11->Enabled=true;

A10->Enabled=false;

B10->Enabled=false;

Edit1->Enabled=true;

Edit1->Text="14";

sv2->Caption="Захвачен пр.1";

}

if (k==12)

{

A12->Enabled=true;

B12->Enabled=true;

A11->Enabled=false;

B11->Enabled=false;

Edit1->Enabled=false;

int B = StrToInt(FlashData->Cells[2][0]);

int buf = StrToInt(Buf1->Caption);

if (buf<B)

f=true;

}

if (k==13)

{

A13->Enabled=true;

B13->Enabled=true;

A12->Enabled=false;

B12->Enabled=false;

Label2->Caption="Заблокирован";

Label2->Color=clRed;

if (f)

FlashData->Cells[1][0]=Buf1->Caption;

//FlashData->Cells[4][0]=Buf2->Caption;

}

if (k==14)

{

A14->Enabled=true;

//B14->Enabled=true;

A13->Enabled=false;

//B13->Enabled=false;

Buf1->Caption=FlashData->Cells[1][0];

}

if (k==15)

{

A15->Enabled=true;

//B15->Enabled=true;

A14->Enabled=false;

sv2->Caption="";

//B14->Enabled=false;

}

if (k==16)

{

A16->Enabled=true;

B14->Enabled=true;

A15->Enabled=false;

FlashData->Cells[2][0]=Buf1->Caption;

B13->Enabled=false;

Label2->Caption="Выполняется";

sv2->Caption="Захвачен пр.2";

Label2->Color=clGreen;

int B=StrToInt(FlashData->Cells[2][0]);

int buf=StrToInt(Buf2->Caption);

if (buf<B)

f=true;

else

f=false;

}

if (k==17)

{

A17->Enabled=true;

B15->Enabled=true;

A16->Enabled=false;

Edit1->Enabled=true;

Edit1->Text="31";

B14->Enabled=false;

if (f)

FlashData->Cells[1][0]=Buf2->Caption;

}

if (k==18)

{

A18->Enabled=true;

B16->Enabled=true;

A17->Enabled=false;

B15->Enabled=false;

Buf1->Caption=Edit1->Text;

Edit1->Enabled=false;

sv2->Caption="";

}

if (k==19)

{

A19->Enabled=true;

B17->Enabled=true;

A18->Enabled=false;

B16->Enabled=false;

FlashData->Cells[4][0]=Buf2->Caption;

sv3->Caption="Захвачен пр.1";

}

if (k==20)

{

A20->Enabled=true;

B18->Enabled=true;

A19->Enabled=false;

B17->Enabled=false;

Label2->Caption="Заблокирован";

Label2->Color=clRed;

B20->Enabled=false;

int C = StrToInt(FlashData->Cells[2][0]);

int buf = StrToInt(Buf1->Caption);

if (buf<C)

f=true;

else

f=false;

}

if (k==21)

{

A21->Enabled=true;

A20->Enabled=false;

if (f)

FlashData->Cells[2][0]=Buf1->Caption;

}

if (k==22)

{

A22->Enabled=true;

A21->Enabled=false;

sv3->Caption="";

}

if (k==23)

{

A22->Enabled=false;

Label2->Caption="Выполняется";

sv3->Caption="Захвачен пр.2";

Label2->Color=clGreen;

B19->Enabled=true;

B18->Enabled=false;

Label1->Caption="Процесс завершён";

Label1->Color=clWindow;

int buf = StrToInt(Buf2->Caption);

int C = StrToInt(FlashData->Cells[2][0]);

if (buf<C)

f=true;

else

f=false;

}

if (k==24)

{

if (f)

FlashData->Cells[2][0]=Buf2->Caption;

B20->Enabled=true;

B19->Enabled=false;

}

if (k==25)

{

B21->Enabled=true;

B20->Enabled=false;

sv3->Caption="";

}

if (k==26)

{

B22->Enabled=true;

B21->Enabled=false;

FlashData->Cells[4][0]=Buf2->Caption;

}

if (k==27)

{

B22->Enabled=false;

Label2->Caption="Процесс завершён";

Label2->Color=clWindow;

}

if (k==28)

{

Timer1->Enabled=false;

Play->Caption="Включить демонстрацию";

k=0;

Offon=false;

for (int i=0; i<5; i++)

FlashData->Cells[i][0]="";

}

}

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

void \_\_fastcall TRealmutex::PlayClick(TObject \*Sender)

{ if (!Offon)

{ Offon=true;

Timer1->Enabled=true;

Label1->Caption="Выполняется";

Label1->Color=clGreen;

Label2->Caption="Выполняется";

Label2->Color=clGreen;

Play->Caption="Завершить демонстрацию";

}

else if (Offon)

{Offon=false;

Timer1->Enabled=false;

k=0;

Label1->Caption="Не запущен";

Label1->Color=clSkyBlue;

Label2->Caption="Не запущен";

Label2->Color=clSkyBlue;

Play->Caption="Включить демонстрацию";

A1->Enabled=false; B1->Enabled=false;

A2->Enabled=false; B2->Enabled=false;

A3->Enabled=false; B3->Enabled=false;

A4->Enabled=false; B4->Enabled=false;

A5->Enabled=false; B5->Enabled=false;

A6->Enabled=false; B6->Enabled=false;

A7->Enabled=false; B7->Enabled=false;

A8->Enabled=false; B8->Enabled=false;

A9->Enabled=false; B9->Enabled=false;

A10->Enabled=false; B10->Enabled=false;

A11->Enabled=false; B11->Enabled=false;

A12->Enabled=false; B12->Enabled=false;

A13->Enabled=false; B13->Enabled=false;

A14->Enabled=false; B14->Enabled=false;

A15->Enabled=false; B15->Enabled=false;

A16->Enabled=false; B16->Enabled=false;

A17->Enabled=false; B17->Enabled=false;

A18->Enabled=false; B18->Enabled=false;

A19->Enabled=false; B19->Enabled=false;

A20->Enabled=false; B20->Enabled=false;

A21->Enabled=false; B21->Enabled=false;

A22->Enabled=false; B22->Enabled=false;

for (int i=0; i<5; i++)

FlashData->Cells[i][0]="";

}

}

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

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

{

Makevr->Visible=true;

Makevr->Show();

}

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

void \_\_fastcall TRealmutex::Button6Click(TObject \*Sender)

{

Realmutex->Close();

Mainmenu->Show();

}

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

Фрагмент текста модуля Unit4.cpp. Полную копию можно посмотреть на диске с приложением

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

#include <vcl.h>

#pragma hdrstop

#include "Unit4.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TMakevr \*Makevr;

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

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

: TForm(Owner)

{

}

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

AnsiString Commarr[15]={"Ввести", "Записать в A", "Записать в B",

"Записать в C", "Захватить A", "Захватить B", "Захватить C",

"Вывести", "Освободить A", "Освободить B", "Освободить C",

"Считать из A", "Считать из B", "Считать из C", ""};

int Procarr[3][10]={14,14,14,14,14,14,14,14,14,14,

14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14};

int i1=0, i2=0, i3=0;

int k=0, k1=0, k2=0, k3=0;

bool l1=false, l2=false, l3=false;

int a=0, b=0, c=0;

void showcom (int ind, int pr, TEdit \*E)

{

if (pr==1)

{

if (ind==0)

E->Text=Commarr[Procarr[0][0]];

else if (ind==1)

E->Text=Commarr[Procarr[0][1]];

else if (ind==2)

E->Text=Commarr[Procarr[0][2]];

else if (ind==3)

E->Text=Commarr[Procarr[0][3]];

else if (ind==4)

E->Text=Commarr[Procarr[0][4]];

else if (ind==5)

E->Text=Commarr[Procarr[0][5]];

else if (ind==6)

E->Text=Commarr[Procarr[0][6]];

else if (ind==7)

E->Text=Commarr[Procarr[0][7]];

else if (ind==8)

E->Text=Commarr[Procarr[0][8]];

else if (ind==9)

E->Text=Commarr[Procarr[0][9]];

}

if (pr==2)

{

if (ind==0)

E->Text=Commarr[Procarr[1][0]];

else if (ind==1)

E->Text=Commarr[Procarr[1][1]];

else if (ind==2)

E->Text=Commarr[Procarr[1][2]];

else if (ind==3)

E->Text=Commarr[Procarr[1][3]];

else if (ind==4)

E->Text=Commarr[Procarr[1][4]];

else if (ind==5)

E->Text=Commarr[Procarr[1][5]];

else if (ind==6)

E->Text=Commarr[Procarr[1][6]];

else if (ind==7)

E->Text=Commarr[Procarr[1][7]];

else if (ind==8)

E->Text=Commarr[Procarr[1][8]];

else if (ind==9)

E->Text=Commarr[Procarr[1][9]];

}

if (pr==3)

{

if (ind==0)

E->Text=Commarr[Procarr[2][0]];

else if (ind==1)

E->Text=Commarr[Procarr[2][1]];

else if (ind==2)

E->Text=Commarr[Procarr[2][2]];

else if (ind==3)

E->Text=Commarr[Procarr[2][3]];

else if (ind==4)

E->Text=Commarr[Procarr[2][4]];

else if (ind==5)

E->Text=Commarr[Procarr[2][5]];

else if (ind==6)

E->Text=Commarr[Procarr[2][6]];

else if (ind==7)

E->Text=Commarr[Procarr[2][7]];

else if (ind==8)

E->Text=Commarr[Procarr[2][8]];

else if (ind==9)

E->Text=Commarr[Procarr[2][9]];

}

}

void f1 (TEdit \*E, TLabel \*b)

{

b->Caption=E->Text;

}

void f2 (TStringGrid \*Tabl, TLabel \*b, bool &p, int &elem, int ind)

{

if (elem==ind || elem==0)

p=true;

else

p=false;

if (p)

Tabl->Cells[0][0]=b->Caption;

}

void f3 (TStringGrid \*Tabl, TLabel \*b, bool &p, int &elem, int ind)

{

if (elem==ind || elem==0)

p=true;

else

p=false;

if (p)

Tabl->Cells[1][0]=b->Caption;

}

void f4 (TStringGrid \*Tabl, TLabel \*b, bool &p, int &elem, int ind)

{

if (elem==ind || elem==0)

p=true;

else

p=false;

if (p)

Tabl->Cells[2][0]=b->Caption;

}

void f5\_6\_7 (int &elem, int ind, bool &p)

{ if (elem==ind || elem==0) {

elem=ind; p=true; }

else

p=false;

}

void f8 (TLabel \*b, TEdit \*E)

{

E->Text=b->Caption;

}

void f9\_10\_11 (int &elem, int ind, bool &p)

{

if (elem==ind || elem==0) {

elem=0; p=true; }

else

p=false;

}

void f12 ( TLabel \*b, TStringGrid \*Tabl, bool &p, int &elem, int ind)

{ if (elem==ind || elem==0) {

b->Caption=Tabl->Cells[0][0];

p=true; }

else

p=false;

}

void f13 ( TLabel \*b, TStringGrid \*Tabl, bool &p, int &elem, int ind)

{ if (elem==ind || elem==0) {

b->Caption=Tabl->Cells[1][0];

p=true; }

else

p=false;

}

void f14 ( TLabel \*b, TStringGrid \*Tabl, bool &p, int &elem, int ind)

{ if (elem==ind || elem==0) {

b->Caption=Tabl->Cells[2][0];

p=true; }

else

p=false;

}

void \_\_fastcall TMakevr::A1Click(TObject \*Sender)

{

if ((Edit1->Text).Pos(".")==0)

{

Edit1->Text=(A1->Caption)+".";

Procarr[0][0]=0;

}

else if ((Edit2->Text).Pos(".")==0)

{

Edit2->Text=(A1->Caption)+".";

Procarr[0][1]=0;

}

else if ((Edit3->Text).Pos(".")==0)

{

Edit3->Text=(A1->Caption)+".";

Procarr[0][2]=0;

}

else if ((Edit4->Text).Pos(".")==0)

{

Edit4->Text=(A1->Caption)+".";

Procarr[0][3]=0;

}

else if ((Edit5->Text).Pos(".")==0)

{

Edit5->Text=(A1->Caption)+".";

Procarr[0][4]=0;

}

else if ((Edit6->Text).Pos(".")==0)

{

Edit6->Text=(A1->Caption)+".";

Procarr[0][5]=0;

}

else if ((Edit7->Text).Pos(".")==0)

{

Edit7->Text=(A1->Caption)+".";

Procarr[0][6]=0;

}

else if ((Edit8->Text).Pos(".")==0)

{

Edit8->Text=(A1->Caption)+".";

Procarr[0][7]=0;

}

else if ((Edit9->Text).Pos(".")==0)

{

Edit9->Text=(A1->Caption)+".";

Procarr[0][8]=0;

}

else if ((Edit10->Text).Pos(".")==0)

{

Edit10->Text=(A1->Caption)+".";

Procarr[0][9]=0;

}

}

//остальную часть кода можно посмотреть на диске с приложением

Текст модуля Unit5.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit5.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TByzantGenerals \*ByzantGenerals;

TStringGrid \*PrTab;

bool Sved[4][4]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

int Zased[4]={0,0,0,0};

int Polog=0;

class General

{

protected:

AnsiString action;

int num;

AnsiString Allpr[4];

bool D;

public:

General(int g)

{

action="Ждать приказ";

D=0;

num=g;

}

void Poluchpr(AnsiString prikaz)

{

action=prikaz;

if (action=="Атаковать")

D=true;

else

D=false;

}

void tallpr(General &X, AnsiString pr)

{

X.Allpr[num]=pr;

}

void showpr(TStringGrid \*T)

{

T->Cells[0][num]=Allpr[0];

T->Cells[1][num]=Allpr[1];

T->Cells[2][num]=Allpr[2];

T->Cells[3][num]=Allpr[3];

}

};

class Commander: public General

{

bool ListGetPr[4];

AnsiString ListPr[4];

public:

void SendPrikaz (AnsiString prikaz, int gen, General &G)

{

General: G.Poluchpr(prikaz);

ListPr[gen]= prikaz;

}

Commander(int g): General ( g)

{

ListPr[0]="Ждать приказ";

ListPr[1]="Ждать приказ";

ListPr[2]="Ждать приказ";

ListPr[3]="Ждать приказ";

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

ListGetPr[i]=0;

}

};

General G1(0);

General G2(1);

General G3(2);

General G4(3);

Commander G(4);

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

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

: TForm(Owner)

{

}

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

void \_\_fastcall TByzantGenerals::BitBtn2Click(TObject \*Sender)

{

Edit1->Text=Edit->Text;

:: G1.Poluchpr(Edit1->Text);

}

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

void \_\_fastcall TByzantGenerals::BitBtn3Click(TObject \*Sender)

{

Edit2->Text=Edit->Text;

:: G2.Poluchpr(Edit2->Text);

}

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

void \_\_fastcall TByzantGenerals::BitBtn4Click(TObject \*Sender)

{

Edit3->Text=Edit->Text;

:: G3.Poluchpr(Edit3->Text);

}

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

void \_\_fastcall TByzantGenerals::BitBtn5Click(TObject \*Sender)

{

Edit4->Text=Edit->Text;

:: G4.Poluchpr(Edit4->Text);

}

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

void \_\_fastcall TByzantGenerals::BitBtn6Click(TObject \*Sender)

{

if ((Edit5->Text).Pos("2"))

:: G1.tallpr(:: G2, Edit1->Text);

if ((Edit5->Text).Pos("3"))

:: G1.tallpr(:: G3, Edit1->Text);

if ((Edit5->Text).Pos("4"))

:: G1.tallpr(:: G4, Edit1->Text);

/\*:: G1.showpr(PrTab); :: G2.showpr(PrTab);

:: G3.showpr(PrTab); :: G4.showpr(PrTab); \*/

}

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

void \_\_fastcall TByzantGenerals::BitBtn7Click(TObject \*Sender)

{

if ((Edit6->Text).Pos("1"))

:: G2.tallpr(:: G1, Edit2->Text);

if ((Edit6->Text).Pos("3"))

:: G2.tallpr(:: G3, Edit2->Text);

if ((Edit6->Text).Pos("4"))

:: G2.tallpr(:: G4, Edit2->Text);

/\*:: G1.showpr(PrTab); :: G2.showpr(PrTab);

:: G3.showpr(PrTab); :: G4.showpr(PrTab); \*/

}

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

void \_\_fastcall TByzantGenerals::BitBtn8Click(TObject \*Sender)

{

if ((Edit7->Text).Pos("1"))

:: G3.tallpr(:: G1, Edit3->Text);

if ((Edit7->Text).Pos("2"))

:: G3.tallpr(:: G2, Edit3->Text);

if ((Edit7->Text).Pos("4"))

:: G3.tallpr(:: G4, Edit3->Text);

/\*:: G1.showpr(PrTab); :: G2.showpr(PrTab);

:: G3.showpr(PrTab); :: G4.showpr(PrTab); \*/

}

void \_\_fastcall TByzantGenerals::BitBtn9Click(TObject \*Sender)

{

if ((Edit8->Text).Pos("1"))

:: G4.tallpr(:: G1, Edit4->Text);

if ((Edit8->Text).Pos("2"))

:: G4.tallpr(:: G2, Edit4->Text);

if ((Edit8->Text).Pos("3"))

:: G4.tallpr(:: G3, Edit4->Text);

/\*:: G1.showpr(PrTab); :: G2.showpr(PrTab);

:: G3.showpr(PrTab); :: G4.showpr(PrTab); \*/

}

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

{ :: G1.tallpr(:: G1, Edit1->Text);

:: G2.tallpr(:: G2, Edit2->Text);

:: G3.tallpr(:: G3, Edit3->Text);

:: G4.tallpr(:: G4, Edit4->Text);

:: G1.showpr(PrTab); :: G2.showpr(PrTab);

:: G3.showpr(PrTab); :: G4.showpr(PrTab);

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

for (int j=0;j<4;j++)

if ((PrTab->Cells[j][i]).Pos("Атаковать"))

Sved[i][j]=true; else Sved[i][j]=false;

}

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

void \_\_fastcall TByzantGenerals::BitBtn1Click(TObject \*Sender)

{

if (Sved[0][0])

Zased[0]=1;

else

Zased[0]=0;

if (Sved[0][1])

Zased[1]=1;

else

Zased[1]=0;

if (Sved[0][2])

Zased[2]=1;

else

Zased[2]=0;

if (Sved[0][3])

Zased[3]=1;

else

Zased[3]=0;

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

if (Sved[i][0] && !(Sved[i+1][0]) || Sved[i+1][0] && !(Sved[i][0]))

Zased[0]=2;

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

if (Sved[i][1] && !(Sved[i+1][1]) || Sved[i+1][1] && !(Sved[i][1]))

Zased[1]=2;

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

if (Sved[i][2] && !(Sved[i+1][2]) || Sved[i+1][2] && !(Sved[i][2]))

Zased[2]=2;

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

if (Sved[i][3] && !(Sved[i+1][3]) || Sved[i+1][3] && !(Sved[i][3]))

Zased[3]=2;

int k=0;

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

if (Zased[i]==2)

k++;

if (k>1)

Polog=4;

if (k==1)

{

Polog=3;

goto m;

}

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

if (Zased[i]==0 && Zased[i+1]!=0 || Zased[i+1]==0 && Zased[i]!=0)

{

Polog=2;

goto m;

}

if (Zased[0]==1 && Zased[1]==1 && Zased[2]==1 && Zased[3]==1)

Polog=1;

else if (Zased[0]==0 && Zased[1]==0 && Zased[2]==0 && Zased[3]==0)

Polog=0;

m:

if (Polog==4) {

Or->Caption="Генералы не смогли договориться.";

Pr->Caption="Предателей несколько. В войсках беспорядки.";

Itog->Caption="Армия взбунтовалась.";

}

if (Polog==3)

{

if (Zased[0]==2)

Pr->Caption="Генерал 1 - предатель.";

else if (Zased[1]==2)

Pr->Caption="Генерал 2 - предатель.";

else if (Zased[2]==2)

Pr->Caption="Генерал 3 - предатель.";

else if (Zased[3]==2)

Pr->Caption="Генерал 4 - предатель.";

if (Zased[0]==1 && Zased[1]==1 && Zased[2]==1 ||

Zased[0]==1 && Zased[2]==1 && Zased[3]==1 ||

Zased[1]==1 && Zased[2]==1 && Zased[3]==1 ||

Zased[0]==1 && Zased[1]==1 && Zased[3]==1)

{

Or->Caption="Решено-атаковать!";

Itog->Caption="Войска победили противника.";

}

else

{

Or->Caption="Решено-отступать!";

Itog->Caption="Войскам пришлось отступить, чтобы сохранить армию.";

}

}

if (Polog==2)

{

Pr->Caption="Главнокомандующий войсками - предатель.";

if (Zased[0]==1 && Zased[1]==1 && Zased[2]==1 ||

Zased[0]==1 && Zased[2]==1 && Zased[3]==1 ||

Zased[1]==1 && Zased[2]==1 && Zased[3]==1 ||

Zased[0]==1 && Zased[1]==1 && Zased[3]==1)

{

Or->Caption="Решено-атаковать!";

Itog->Caption="Войска победили противника.";

}

else

{

Or->Caption="Решено-отступать!";

Itog->Caption="Войскам пришлось отступить, чтобы сохранить армию.";

}

}

if (Polog==1)

{

Pr->Caption="Все генералы верны.";

Or->Caption="Решено-атаковать!";

Itog->Caption="Войска победили противника.";

}

if (Polog==0)

{

Pr->Caption="Все генералы верны.";

Or->Caption="Решено-отступать!";

Itog->Caption="Войскам пришлось отступить, чтобы сохранить армию.";

}

if ((Edit1->Text).Pos("Ждать приказ") || (Edit2->Text).Pos("Ждать приказ")

|| (Edit3->Text).Pos("Ждать приказ") || (Edit4->Text).Pos("Ждать приказ"))

{

Or->Caption="Ждать решение.";

Pr->Caption="Неопределены.";

Itog->Caption="Нужно отдать приказы всем генералам и дать им обменяться сведениями.";

}

if (!((Edit1->Text).Pos("Атаковать")) && !((Edit1->Text).Pos("Отступать"))

|| !((Edit2->Text).Pos("Атаковать")) && !((Edit2->Text).Pos("Отступать"))

|| !((Edit3->Text).Pos("Атаковать")) && !((Edit3->Text).Pos("Отступать"))

|| !((Edit4->Text).Pos("Атаковать")) && !((Edit4->Text).Pos("Отступать")))

{ Or->Caption="Ждать решение.";

Pr->Caption="Неопределены.";

Itog->Caption="Нужно отдать приказы всем генералам и дать им обменяться сведениями.";

}

}

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

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

{

ByzantGenerals->Close();

Mainmenu->Show();

}

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

Текст модуля Unit6.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit6.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TClockSyn \*ClockSyn;

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

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

: TForm(Owner)

{

}

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

int E;

int D;

int k=0, k1=0;

int TimClock1[3];

int TimClock2[3];

int Ta1=0; int Ta2=0; int Tb1=0; int Tb2=0;

void \_\_fastcall TClockSyn::BitBtn1Click(TObject \*Sender)

{

k=0; k1=0;

m1->Visible=false; m2->Visible=false;

m3->Visible=false; m4->Visible=false;

m5->Visible=false; m6->Visible=false;

TimClock1[0]=StrToInt(Edit1->Text);

TimClock1[1]=StrToInt(Edit2->Text);

TimClock2[0]=StrToInt(Edit3->Text);

TimClock2[1]=StrToInt(Edit4->Text);

D=StrToInt(Edit5->Text);

Timer2->Enabled=true;

}

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

void \_\_fastcall TClockSyn::Timer1Timer(TObject \*Sender)

{

TimClock1[0]=StrToInt(Edit1->Text);

TimClock1[1]=StrToInt(Edit2->Text);

TimClock2[0]=StrToInt(Edit3->Text);

TimClock2[1]=StrToInt(Edit4->Text);

TimClock1[1]++; TimClock2[1]++;

if (TimClock1[1]>=60)

{

TimClock1[1]=0;

TimClock1[0]++;

if (TimClock1[0]>=24)

TimClock1[0]=0;

}

if (TimClock2[1]>=60)

{

TimClock2[1]=0;

TimClock2[0]++;

if (TimClock2[0]>=24)

TimClock2[0]=0;

}

Edit1->Text=IntToStr(TimClock1[0]);

Edit2->Text=IntToStr(TimClock1[1]);

Edit3->Text=IntToStr(TimClock2[0]);

Edit4->Text=IntToStr(TimClock2[1]);

}

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

void \_\_fastcall TClockSyn::Timer2Timer(TObject \*Sender)

{

k++;

if (k==D)

{

k=0;

k1++;

}

if (k1==0) {

Button1->Enabled=true;

m1->Visible=true;

Ta1=TimClock1[0]\*60+TimClock1[1];

}

else if (k1==1)

{

Button1->Enabled=false;

Button2->Enabled=true;

m2->Visible=true;

m3->Visible=true;

Tb1=TimClock2[0]\*60+TimClock2[1];

}

else if (k1==2)

{

Button2->Enabled=false;

Button3->Enabled=true;

m4->Visible=true;

m5->Visible=true;

Ta2=TimClock1[0]\*60+TimClock1[1];

}

else if (k1==3)

{

Button3->Enabled=false;

m5->Visible=true;

m6->Visible=true;

E=(2\*Tb1-Ta2-Ta1)/2;

int l=TimClock2[0]\*60+TimClock2[1];

l=l-E;

TimClock2[0]=l/60;

TimClock2[1]=l%60;

Edit3->Text=IntToStr(TimClock2[0]);

Edit4->Text=IntToStr(TimClock2[1]);

Timer2->Enabled=false;

}

}

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

void \_\_fastcall TClockSyn::Button4Click(TObject \*Sender)

{

Mainmenu->Show();

Mainmenu->Visible=true;

Timer1->Enabled=false;

Timer2->Enabled=false;

ClockSyn->Close();

}

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

void \_\_fastcall TClockSyn::Image1Click(TObject \*Sender)

{

Timer1->Enabled=true;

}

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

Текст модуля Unit7.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit7.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm7 \*Form7;

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

\_\_fastcall TForm7::TForm7(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcall TForm7::Button4Click(TObject \*Sender)

{

Mainmenu->Visible=true;

Mainmenu->Show();

Form7->Close();

}

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

Текст модуля Unit8.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit8.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm8 \*Form8;

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

\_\_fastcall TForm8::TForm8(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcall TForm8::Button4Click(TObject \*Sender)

{

Form8->Visible=false;

Mainmenu->Show();

Mainmenu->Visible=true;

Form8->Close();

}

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

Текст модуля Unit9.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit9.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm9 \*Form9;

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

\_\_fastcall TForm9::TForm9(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcall TForm9::Button4Click(TObject \*Sender)

{

Mainmenu->Visible=true;

Mainmenu->Show();

Form9->Visible=false;

Form9->Close();

}

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

Текст модуля Unit10.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit10.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm10 \*Form10;

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

\_\_fastcall TForm10::TForm10(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcall TForm10::Button4Click(TObject \*Sender)

{

Mainmenu->Show();

Mainmenu->Visible=true;

Form10->Visible=false;

Form10->Close();

}

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

Текст модуля Unit11.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit11.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm11 \*Form11;

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

\_\_fastcall TForm11::TForm11(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcall TForm11::Button4Click(TObject \*Sender)

{

Mainmenu->Show();

Mainmenu->Visible=true;

Form11->Visible=false;

Form11->Close();

}

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

Текст модуля Unit12.cpp

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

#include <vcl.h>

#pragma hdrstop

#include "Unit12.h"

#include "Unit2.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm12 \*Form12;

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

\_\_fastcall TForm12::TForm12(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcall TForm12::Button4Click(TObject \*Sender)

{

Mainmenu->Show();

Mainmenu->Visible=true;

Form12->Visible=false;

Form12->Close();

}

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