Министерство образования и науки Российской Федерации

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Специальность (направление, профиль) программная инженерия

Курсовой проект

защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Б. Целебровский

(подпись руководителя проекта)(инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Курсовой проект

Моделирование системы виртуальной памяти с использованием одноуровневой таблицы страниц с использованием алгоритма замещения страниц «Часы»

(тема курсового проекта)

Пояснительная записка

по дисциплине Курсовая работа по операционным системам

\_\_\_\_\_\_\_КП 09.03.04.03.000 ПЗ\_\_\_\_\_\_\_

(обозначение документа)

Студент группы ПИ-02 Козловский\_Н.Н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_30.05.2023

(фамилия, имя, отчество) (подпись) (дата)

Руководитель проекта ассистент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Б. Целебровский

(должность, ученое звание) (подпись) (инициалы, фамилия)

БАРНАУЛ 2023

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный технический университет

имени И.И. Ползунова»

Факультет информационных технологий

Кафедра «Прикладная математика»

З А Д А Н И Е

на курсовой проект по дисциплине «Курсовая работа по операционным системам»

студенту группы ПИ-02 Козловскому Никите Николаевичу

Моделирование системы виртуальной памяти с использованием одноуровневой таблицы страниц с использованием алгоритма замещения страниц «Часы»

Календарный план работы:

|  |  |  |
| --- | --- | --- |
| № этапа | Содержание этапа | Недели семестра |
| 1 | Получение задания | 1 |
| 2 | Описание предметной области и постановка задачи | 2 |
| 3 | Проектирование программы | 3-4 |
| 4 | Реализация программы | 5-13 |
| 5 | Оформление пояснительной записки | 14 |
| 6 | Защита курсового проекта | 15-16 |

Руководитель проекта ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Б. Целебровский подпись

Дата выдачи задания «11» февраля 2023 г.

число месяц год

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Козловский Н.Н.

подпись

Оглавление

[Введение 4](#_Toc134481861)

[1 Обзор предметной области и постановка задачи 5](#_Toc134481862)

[1.1 Обзор предметной области 5](#_Toc134481863)

[1.2 Постановка задачи 6](#_Toc134481864)

[1.3 Требования к функциональности программы 7](#_Toc134481865)

[2 Проект программного продукта 9](#_Toc134481866)

[3. Описание программного продукта 12](#_Toc134481867)

# Введение

На данный момент не смотря на преобладание высокоуровневых решений, увеличение мощности вычислительных систем, в купе с их повышающейся ёмкостью, для любого спектра задач, остаётся актуальной проблема оптимального распределения и использования памяти.

Цель данной работы: Моделирование системы виртуальной памяти с использованием одноуровневой таблицы страниц с использованием алгоритма замещения страниц «Часы»

Главная задача: написать программу с графическим интерфейсом, позволяющую наглядно изучить работу алгоритма «Часы» для виртуальной памяти.

# Обзор предметной области и постановка задачи

## **Обзор предметной области**

**Оптимальный страничный алгоритм**

Данный алгоритм выгружает страницу с наибольшей меткой, обозначающей, в количестве операций, задержку (простой) в обращении к этой странице. Чем больше операций необходимо ждать перед следующим обращением, тем меньше необходимость держать страницу в памяти.

К сожалению, данный алгоритм не применим в реальных системах, т.к. ОС не знает о точном количестве и конкретном времени обращения к соответствующей странице.

**Биты R и M**

Чтобы дать возможность операционной системе собирать полезные статистические данные о том, какие страницы используются, а какие — нет, большинство компьютеров с виртуальной памятью поддерживают два статусных бита, связанных с каждой страницей. Бит R (Referenced — обращения) устанавливается всякий раз, когда происходит обращение к странице (чтение или запись). Бит М (Modified — изменение) устанавливается, когда страница записывается (то есть изменяется). Биты содержатся в каждом элементе таблицы страниц, как показано на рис. 2.10. Важно реализовать обновление этих битов при каждом обращении к памяти, поэтому необходимо, чтобы они задавались аппаратно. Если однажды бит был установлен в 1, то он остается равным 1 до тех пор, пока операционная система программно не вернет его в состояние 0.

Если аппаратное обеспечение не поддерживает эти биты, их можно смоделировать следующим образом. Когда процесс запускается, все его записи в таблице страниц помечаются как отсутствующие в памяти. Как только происходит обращение к странице, происходит страничное прерывание. Затем операционная система устанавливает бит R (в своих внутренних таблицах); изменяет запись в таблице страниц, чтобы она указывала на корректную страницу с режимом READ ONLY (только для чтения), и перезапускает команду. Если страница позднее записывается, происходит другое страничное прерывание, позволяющее операционной системе установить бит М и изменить состояние страницы на READ/WRITE (чтение/запись).

Биты R и M могут использоваться для построения простого алгоритма замещения страниц, описанного ниже. Когда процесс запускается, оба страничных бита для всех его страниц операционной системой установлены на 0. Периодически (например, при каждом прерывании по таймеру) бит R очищается, чтобы отличить страницы, к которым давно не происходило обращения от тех, на которые были ссылки.

Когда возникает страничное прерывание, операционная система проверяет все страницы и делит их на четыре категории на основании текущих значений битов R и M:

Класс 0: не было обращений и изменений.

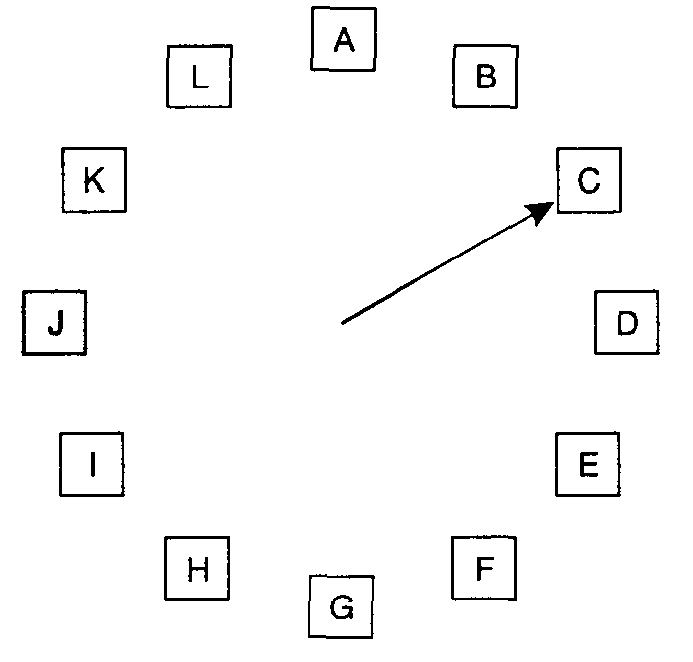
Класс 1: не было обращений, страница изменена.

Класс 2: было обращение, страница не изменена.

Класс 3: произошло и обращение, и изменение.

**Алгоритм «Часы»**

Все страничные блоки хранятся в кольцевом списке в форме часов, как показано на рис. Стрелка указывает на старейшую страницу.



Когда происходит страничное прерывание, проверяется та страница, на которую направлена стрелка. Если ее бит R равен 0, страница выгружается, на ее место в часовой круг встает новая страница, а стрелка сдвигается вперед на одну позицию. Если бит R равен 1, то он сбрасывается, стрелка перемещается к следующей странице. Этот процесс повторяется до тех пор, пока не находится та страница, у которой бит R = 0.

## **Постановка задачи**

Реализация ПО для демонстрации работы алгоритма «Часы»

* Реализация модели виртуальной памяти, путём реализации её основного атрибута: **Страницы**. В состав страницы должен входить:
  + Бит обращения R необходимый, для реализации алгоритма.
  + Индекс в таблице процесса: виртуальная память
  + Индекс в общей таблице памяти: физическая память
* Для обработки страниц должен быть реализован **Процесс**, являющийся моделью виртуальной памяти. Функционал **Процесса** должен включать:
  + Страничное прерывание, определяющее поведение алгоритма замещения страниц.
  + Добавление нового страницы в таблицу, из физической памяти
  + Вывод результатов работы прерывания и состояния процесса
* В качестве пространства существования Процесса, необходимо создать модель **Физической памяти**, основными задачами которой являются:
  + Создание процесса
  + Передача доступа процессу к уникальным страницам, для избежания повторов во внутренней таблице.
  + Один тик работы памяти, инкапсулирующий весь функционал физической памяти и виртуальной памяти.
  + Выводить состояние всей памяти, включая реализуемый внутри неё процесс

## **Требования к функциональности программы**

* Для простого взаимодействия с моделью памяти, через интуитивно понятный вывод и интерфейс настройки, необходимо создать **Окно**, выполняющее роли:
  + Организация результатов работы модели
  + Вывода результатов работы модели
  + Запуск модели с заданным количеством страниц
  + Перемещение модели по тикам памяти, с соответствующим отображением

# Проект программного продукта

## **Математическая модель**

**Окно**

Пользователь вводит количество страниц в соответствующей строке и нажимает кнопку создать. Вызванный обработчик события запускает соответствующие функции создания модели, начиная с вышестоящего в классовой иерархии: **физической памяти.**

**Физическая память**

Конструктор создаёт экземпляр Физической памяти с выставленным количеством страниц. В соответствии с последовательностью создания, каждому экземпляру страницы присваивается индекс, характеризующий расположение страницы в физической памяти: этот индекс не изменяется на протяжении всей работы с моделью.

Далее создаётся процесс, находящийся в экземпляре используемой физической памяти. Процесс создания разделён на три этапа:

* Создание последовательности физических индексов, идентичных хранимым в страницах. Эта последовательность уникальных элементов, не используемых в процессе. Наличие такой последовательности позволяет разделять элементы, без их удаления.
* Создание случайной уникальной последовательности страниц на основе созданной ранее последовательности. При добавлении страницы в последовательность соответствующий элемент из неё удаляется.
* Создание процесса путём передачи ему последовательности в качестве параметра

**Процесс**

Т.к. это модель работы конкретного алгоритма, то все функции процесса связаны с основной задачей: алгоритмом замещения страниц. Данный алгоритм реализован в страничном прерывании, поэтому до соответствующей цепочки вызовов, процесс является не более, чем контейнером для данных.

Страницы внутри организованны таким образом, что их последовательность упорядочена в соответствии с индексом внутренней таблицы процесса, моделирующей виртуальную память. Последовательность индексов устанавливается при добавлении в процесс и может изменяться в отличии от физической памяти.

**Окно**

Пользователь нажимает соответствующую кнопку следующий Тик, что приводит модель в действие. После череды вызовов и обработок, результатом вывода становятся данные по модели в трёх колонках: **До прерывания**, **После прерывания**, **Ход прерывания.** Так же увеличивается счётчик тиков напротив нажатой кнопки.

**Физическая память**

Вызывается страничное прерывание от экземпляра процесса. Результатом становится страница, удалённая из **внутренней** таблицы процесса, в следствии работы алгоритма «Часы». Происходит проверка на возврат ошибки.

Если проверка пройдена успешно, запускается процесс замещения удалённой страницы:

* Удаление возвращённой страницы из физической памяти. Все страницы в данной работе приняты за неизменяемые, т.к. бит изменения M даже не был реализован.
* Проверка на наличие уникальных элементов. Проверяется длинна контейнера уникальных физических индексов. Если длинна не ровна 0, значит остались не использованные процессом страницы в памяти.
* Случайный выбор физического индекса из контейнера с уникальными индексами
* Поиск соответствующей страницы в физической памяти
* При наличии соответствия, добавление элемента в внутреннюю таблицу процесса

**Процесс**

Работа с страницами в цепочке вызовов для процесса идёт в последовательности: страничное прерывание, замещение удалённого элемента.

**Страничное прерывание**

Страничное прерывание реализует функционал алгоритма «Часы», путём последовательного выполнения двух операций: доступа по маске и обработке доступа.

Маска доступа – строка, не превышающая длинной количества страниц во внутренней таблице. Символы, содержащиеся в маске доступа, соответствуют своими индексами индексам виртуальной памяти. Символы в строке могут быть только **0** и **1**, т.к. маска доступа генерируется случайным образом. В соответствии с маской, а именно 1 производится установка бита R в соответствующих процессах.

Результатом такого взаимодействия становится маска взаимодействия. Эта маска используется для вывода результата функции, и характеризуется состояние битов R во всех страницах внутренней таблицы до обработки алгоритмом. Вторая маска получается путём наслоения уже установленных битов R и битов R в текущем обращении, т.к. снимаются биты R только алгоритмом замещения.

После всех предварительных взаимодействий начинается обработка алгоритмом. Т.к. результатом работы будет возврат из прерывания, алгоритм выполнен в виде бесконечно цикла. Цикл постоянно проходит все элементы коллекции, начиная с элемента, на который указывает индекс, хранящийся в «стрелке часов». Поэтому:

* Проверка значения «стрелки часов» на превышение количества страниц в строке. Это значит, что стрелка достигла конца коллекции, и для за имитации закольцованности, обязана отправится в её начало
* Если бит R у страницы, на которую указывает стрелка равен 1, то снимаем его и двигаем стрелку дальше
* Если равен 0, то удаляем этот элемент из внутренней таблицы страниц. При отсутствии внешних замещений, коллекция сдвинется таким образом, что стрелка будет указывать уже на следующий элемент.

**Замещение удалённого элемента**

При продолжении последовательности взаимодействий, физическая память передаст в функцию добавления элемента, уникальную страницу. Функция в свою очередь добавит полученный элемент в коллекцию на место, куда указывала «стрелка часов» при этом определив индекс виртуальной памяти. Т.к. произойдёт смещение коллекции, «стрелка часов будет перемещена на один элемент вперёд».

**Окно**

* **window** – Окно приложения
* **Frame\_of\_output** – Область вывода состояния модели
* **Frame\_of\_input** – Область ввода параметров модели
* **Memory** – Физическая память. Реализация модели

**Физическая память**

* **Pages\_in\_memory** – Коллекция страниц в физической памяти
* **Index\_pages** – Индексы физической памяти уникальных страниц.
* **Process\_in\_memory –** Процесс в физической памяти

**Процесс**

* **Arrow** – Стрелка часов
* **Pages** – Внутренняя таблица [Виртуальная память]
* **Access\_mask** – Маска доступа к таблицам
* **RPages** – Маска результата взаимодействия с таблицами
* **History\_Access** – История работы алгоритма замещения страниц

**Страница**

* **R –** Бит обращения
* **physical –** Индекс физической памяти
* **virtual\_index –** Индекс виртуальной памяти

## **Диаграмма классов**

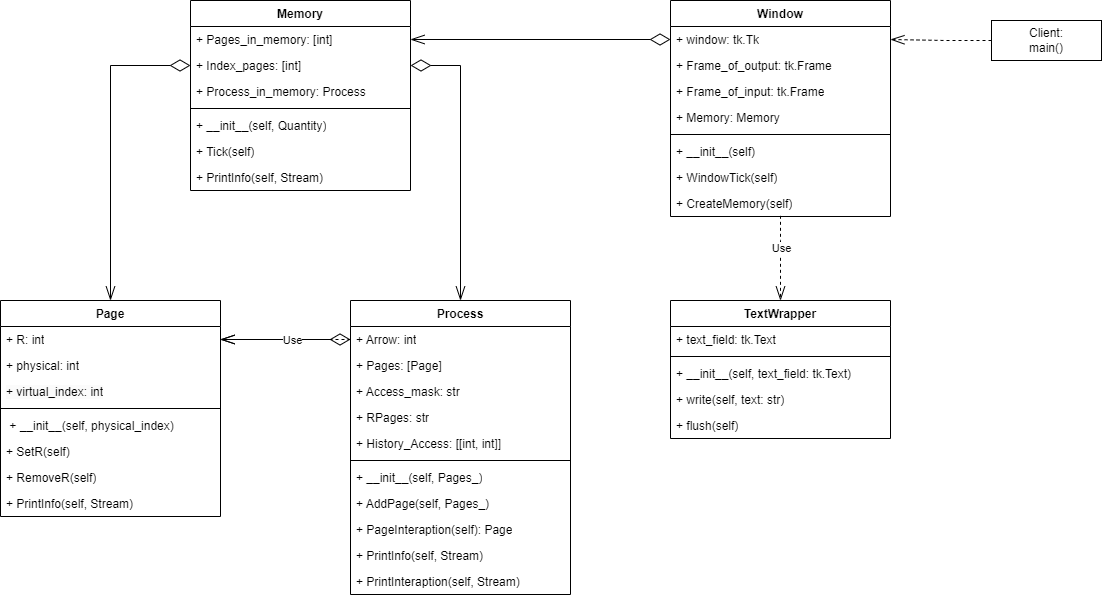


Рисунок 1 - Диаграмма классов

# Описание программного продукта

## **Выбор средств реализации**

Данная модель была реализована на языке Python с использованием среды разработки программного обеспечения Spyder. Python — язык программирования высокого уровня общего назначения. Обладает типизацией динамического строгого характера. Имеет автоматическое управление памятью, за счет чего осуществляется повышение производительности контента, написанного на нем.

Использовались библиотеки:

* **Tkinter** - это графическая библиотека, позволяющая создавать программы с оконным интерфейсом. Эта библиотека является интерфейсом к популярному языку программирования и инструменту создания графических приложений tcl/tk. Tkinter, как и tcl/tk, является кроссплатформенной библиотекой и может быть использована в большинстве распространённых операционных систем
* Модуль **random** предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.

## **Описание классов**

**Окно: Window**

**Поля:**

* **window** – Окно приложения
* **Frame\_of\_output** – Область вывода состояния модели
* **Frame\_of\_input** – Область ввода параметров модели
* **Memory** – Физическая память. Реализация модели

**Методы:**

* **\_\_init\_\_** -конструктор, формирующий окно приложения
* **WindowTick –** обработчик события: нажатие кнопки «Следующий тик»
* **CreateMemory** – обработчик события: нажатие кнопки «Создать»

**Физическая память: Memory**

**Поля:**

* **Pages\_in\_memory** – Коллекция страниц в физической памяти
* **Index\_pages** – Индексы физической памяти уникальных страниц.
* **Process\_in\_memory –** Процесс в физической памяти

**Методы:**

* **\_\_init\_\_** - конструктор, формирующий модель
* **Tick** – моделирование одного тика памяти
* **PrintInfo** –печать состояния модели

**Процесс: Process**

**Поля:**

* **Arrow** – Стрелка часов
* **Pages** – Внутренняя таблица [Виртуальная память]
* **Access\_mask** – Маска доступа к таблицам
* **RPages** – Маска результата взаимодействия с таблицами
* **History\_Access** – История работы алгоритма замещения страниц

**Методы:**

* **\_\_init\_\_ -** конструктор, формирующий процесс
* **AddPage** – добавление страницы на место «стрелки»
* **PageInteraption –** страничное прерывание
* **PrintInfo –** печать состояния процесса

**Страница: Page**

**Поля:**

* **R –** Бит обращения
* **physical –** Индекс физической памяти
* **virtual\_index –** Индекс виртуальной памяти

**Методы:**

* **\_\_init\_\_** - конструктор, формирующий страницу
* **SetR** – установка бита обращения R
* **RemoveR** – снятие бита обращения R
* **PrintInfo** – печать результата

**Поток вывода: TextWrapper**

**Поля:**

* **text\_field –** Декорируемый поток вывода

**Методы:**

* **\_\_init\_\_** - конструктор, формирующий поток вывода
* **write** – переопределённый метод записи stdout
* **flush** – переопределённый метод очистки потока

# Заключение

Была разработана программа, которая обладает следующим функционалом:

* Ввод параметров моделирования
  + Кол-во создаваемых страниц
* Процесс моделирования на основе введенных параметров
* Вывод результатов моделирования в графическое окно
* Пошаговое моделирование

Возможны дальнейшие усовершенствования программы:

* Добавление нескольких процессов
* Добавление создания процесса с заданным количеством страниц процессе
* Обращение процесса по задаваемой на каждом шаге маске доступа

# Список использованных источников

1. Википедия[Электронный ресурс]. – Режим доступа: https://wikichi.ru/wiki/Rule\_30, свободный.
2. Сайт CyberForum [Электронный ресурс]. Режим доступа: https://www.cyberforum.ru/, свободный.
3. Лутц, Марк. Л86 Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019.—832 с. : ил. — Парад, тит. англ. ISBN 978-5 907144-52-1 (рус., том 1) ISBN 978-5-907144-51-4 (рус., многотом.)
4. Доусон М. Программируем на Python. - СПб.: Питер, 2014. - 416 с.: ил.
5. Бейдер Д. Б41 Чистый Python. Тонкости программирования для профи. — СПб.: Питер, 2018. — 288 с.: ил. — (Серия «Библиотека программиста»). ISBN 978-5-4461-0803-9

# Приложение A. Снимки экранных форм пользовательского интерфейса

При запуске программы появляется графический интерфейс с окном, представленным на рисунке 2.

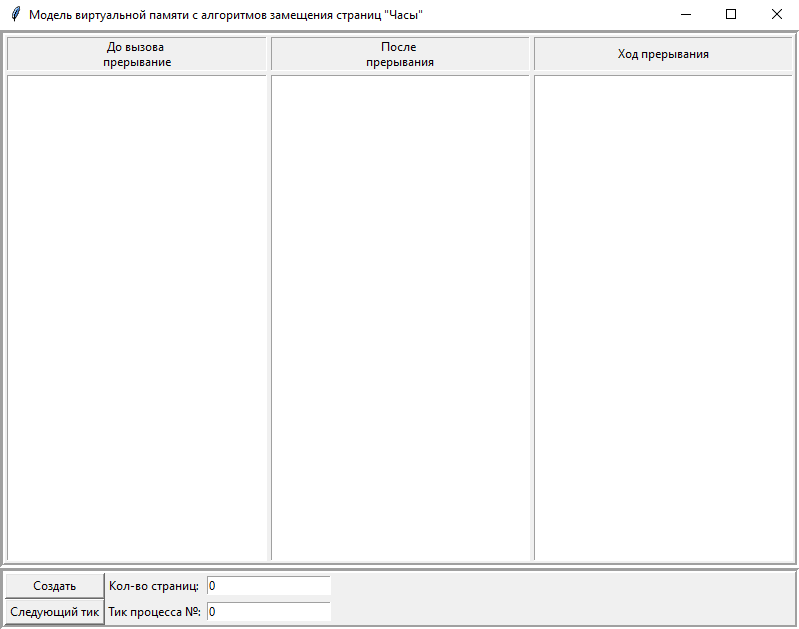


Рисунок 2 - При старте программы

Далее можно ввести кол-во страниц в физической памяти



Рисунок 3 - Параметры моделирования

При нажатии на кнопку «создать», появляется соответствующий экземпляр модели, как на рисунке 4.

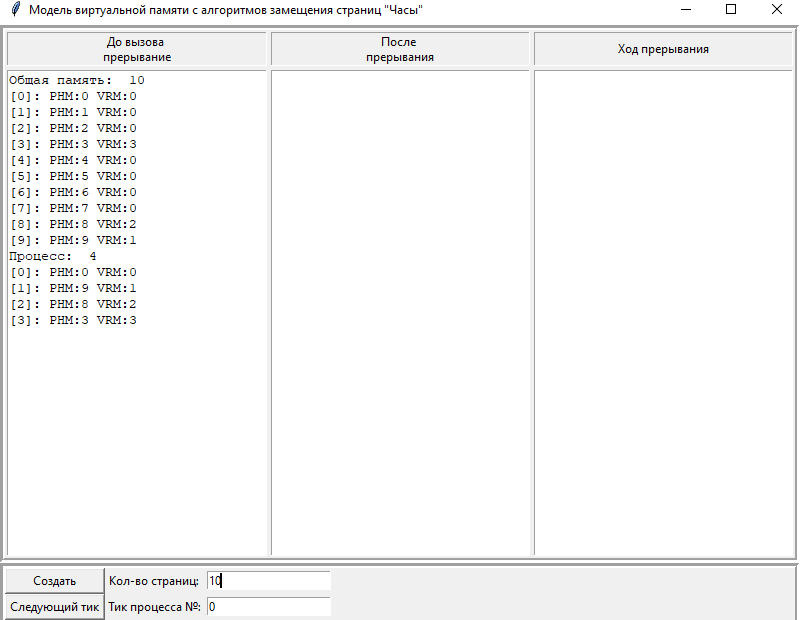


Рисунок 4 – Начало моделирования

Процесс моделирования продолжается пошагово. Каждый шаг модели инициирует пользователь нажатием соответствующей кнопки «Следующий тик». На против нажатой кнопки есть счётчик, который показывает текущий шаг с момента начала моделирования. В результате в окнах «после прерывания» и «ход прерывания» изображается информация о модели, соответствующая текущему шагу.

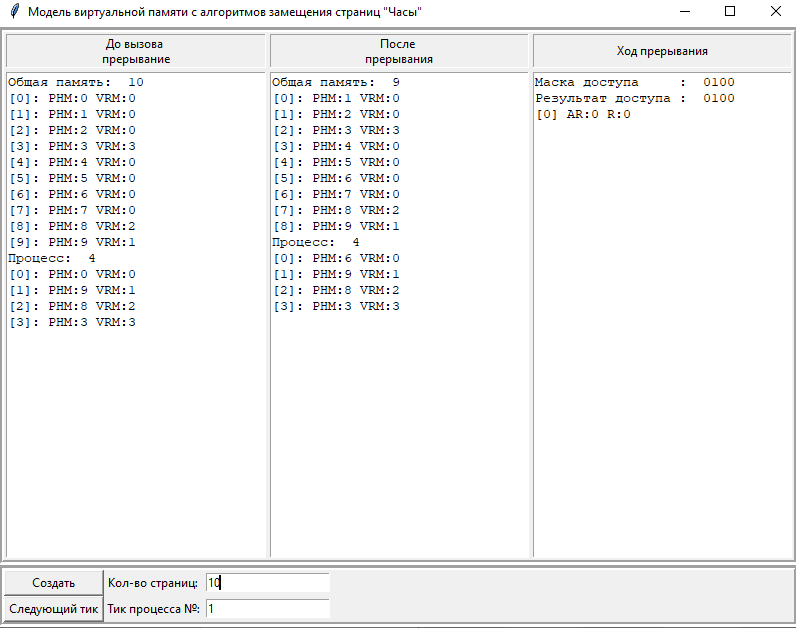


Рисунок 5 – Следующий шаг

Логическим завершение работы модели является опустошение общей памяти. Но программа не оповещает пользователя об этом. Нажатия кнопки «Следующий шаг» перестают оказывать какое-либо воздействие на модель. Рис.6

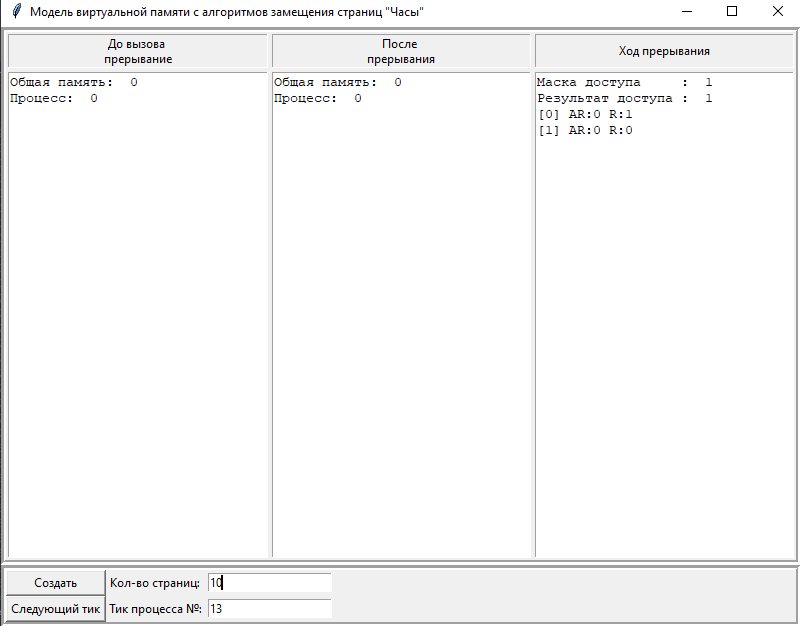


Рисунок 6 – Логическое завершение работы модели

Однако пользователь может пересоздать последовательность на любом этапе, просто нажав кнопку «Создать» с новыми данными. Рис. 7

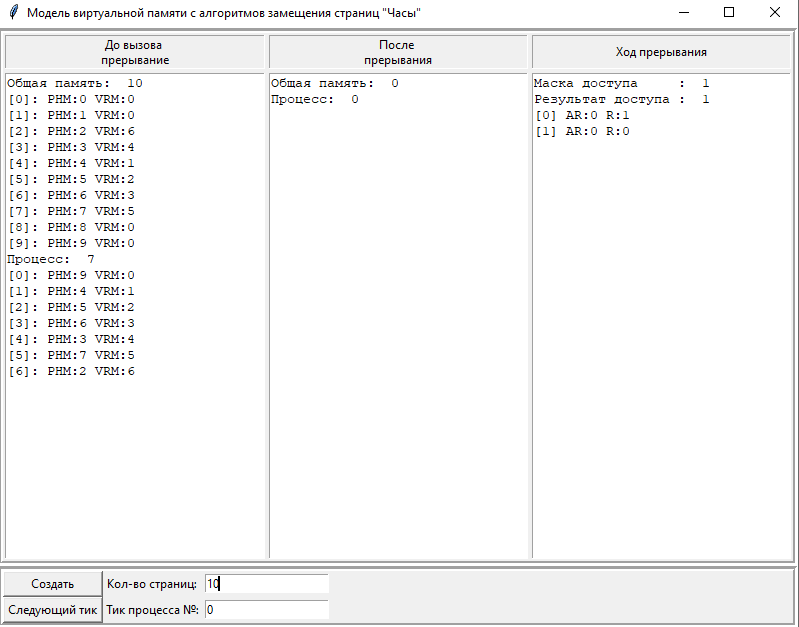


Рисунок 7 – Создание новой последовательности после логического завершение работы модели

# Приложение Б. Исходный код

Исходный код: <https://github.com/BF8DF7D/Virtual-memory-model-with-Clock-page-replacement-algorithms>

