

**Лабораторная работа на тему**  
**«Моделирование работы КЭШ-памяти»**  
**Левасюк Денис, ИВТ-18-2**

Интерфейс программы:

**Моделирование работы КЭШ памяти**

КЭШ память	Основная память
-1 0 0 0 0	1890 2167 6602 7111
-1 0 0 0 0	7636 8549 5893 2200
-1 0 0 0 0	2160 9612 3449 7348
-1 0 0 0 0	8754 2087 1610 5130
-1 0 0 0 0	5373 2738 9206 7665
-1 0 0 0 0	5081 6688 3203 5720
-1 0 0 0 0	2087 7455 3837 2932
-1 0 0 0 0	6888 9098 8411 1334
-1 0 0 0 0	9083 9361 7763 7198
-1 0 0 0 0	8023 9956 6026 1223
	2737 7715 1035 6725
	2801 5623 8327 7338
	6491 1481 8747 7023
	2718 2677 8995 5200
	8962 1881 6023 1181
	5711 3960 5311 3060
	2200 3114 8263 9392
	4323 4411 7106 5415
	6711 8982 5927 7365
	6653 6477 3646 6463

**Сегмент:**

**Строка:**

**Элемент:**

**Найти**

**Элемент:**

**Изменить**

**Время:**

**Строка:**

**Элемент:**

**Статус:** Файл основной памяти заполнен произвольными значениями

Проект на GitHub:

<https://github.com/LevasyukDY/ECM/tree/master/%D0%9A%D0%AD%D0%A8%20-%D0%A0%D0%B0%D0%B1%D0%BE%D1%82%D0%B0>

Загрузка элемента из Основной памяти:

Моделирование работы КЭШ памяти

КЭШ память	Основная память	Сегмент:	Строка:	Элемент:
-1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0 2 8962 1881 6023 1181 -1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0	1890 2167 6602 7111 7636 8549 5893 2200 2160 9612 3449 7348 8754 2087 1610 5130 5373 2738 9206 7665 5081 6688 3203 5720 2087 7455 3837 2932 6888 9098 8411 1334 9083 9361 7763 7198 8023 9956 6026 1223  2737 7715 1035 6725 2801 5623 8327 7338 6491 1481 8747 7023 2718 2677 8995 5200 8962 1881 6023 1181 5711 3960 5311 3060 2200 3114 8263 9392 4323 4411 7106 5415 6711 8982 5927 7365	2	5	3
		<b>Найти</b>		
		Элемент:		
		<b>Изменить</b>		
		Время:	00:00:00.0009009	
		Строка:	8962 1881 6023 1181	
		Элемент:	6023	
<b>Статус:</b> Элемент загружен из Основной памяти				

Моделирование работы КЭШ памяти

КЭШ память	Основная память	Сегмент:	Строка:	Элемент:
-1 0 0 0 0 -1 0 0 0 0 2 6491 1481 8747 7023 -1 0 0 0 0 2 8962 1881 6023 1181 -1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0 -1 0 0 0 0	1890 2167 6602 7111 7636 8549 5893 2200 2160 9612 3449 7348 8754 2087 1610 5130 5373 2738 9206 7665 5081 6688 3203 5720 2087 7455 3837 2932 6888 9098 8411 1334 9083 9361 7763 7198 8023 9956 6026 1223  2737 7715 1035 6725 2801 5623 8327 7338 6491 1481 8747 7023 2718 2677 8995 5200 8962 1881 6023 1181 5711 3960 5311 3060 2200 3114 8263 9392 4323 4411 7106 5415 6711 8982 5927 7365	2	3	4
		<b>Найти</b>		
		Элемент:		
		<b>Изменить</b>		
		Время:	00:00:00.0001971	
		Строка:	6491 1481 8747 7023	
		Элемент:	7023	
<b>Статус:</b> Элемент загружен из Основной памяти				

Изменение значения элемента, загруженного из КЭШ-памяти:

Моделирование работы КЭШ памяти

КЭШ память	Основная память
-1 0 0 0 0	1890 2167 6602 7111
-1 0 0 0 0	7636 8549 5893 2200
2 6491 1481 8747 5555	2160 9612 3449 7348
-1 0 0 0 0	8754 2087 1610 5130
2 8962 1881 6023 1181	5373 2738 9206 7665
-1 0 0 0 0	5081 6688 3203 5720
-1 0 0 0 0	2087 7455 3837 2932
-1 0 0 0 0	6888 9098 8411 1334
-1 0 0 0 0	9083 9361 7763 7198
-1 0 0 0 0	8023 9956 6026 1223
	2737 7715 1035 6725
	2801 5623 8327 7338
	6491 1481 8747 7023
	2718 2677 8995 5200
	8962 1881 6023 1181
	5711 3960 5311 3060
	2200 3114 8263 9392
	4323 4411 7106 5415
	6711 8982 5927 7365
	8853 6477 3046 6463

Сегмент: 2  
Строка: 3  
Элемент: 4  
Найти  
Элемент: 5555  
Изменить  
Время: 00:00:00.0002322  
Строка: 6491 1481 8747 5555  
Элемент: 5555  
Статус: Элемент загружен из КЭШ памяти

Загрузка строки из Основной памяти при том, что место в КЭШ-памяти, занятое другой строкой, освободилось:

Моделирование работы КЭШ памяти

КЭШ память	Основная память
-1 0 0 0 0	1890 2167 6602 7111
-1 0 0 0 0	7636 8549 5893 2200
1 2160 9612 3449 7348	2160 9612 3449 7348
-1 0 0 0 0	8754 2087 1610 5130
2 8962 1881 6023 1181	5373 2738 9206 7665
-1 0 0 0 0	5081 6688 3203 5720
-1 0 0 0 0	2087 7455 3837 2932
-1 0 0 0 0	6888 9098 8411 1334
-1 0 0 0 0	9083 9361 7763 7198
-1 0 0 0 0	8023 9956 6026 1223
	2737 7715 1035 6725
	2801 5623 8327 7338
	6491 1481 8747 5555
	2718 2677 8995 5200
	8962 1881 6023 1181
	5711 3960 5311 3060
	2200 3114 8263 9392
	4323 4411 7106 5415
	6711 8982 5927 7365
	8853 6477 3046 6463

Сегмент: 1  
Строка: 3  
Элемент: 1  
Найти  
Элемент: 5555  
Изменить  
Время: 00:00:00.0007010  
Строка: 2160 9612 3449 7348  
Элемент: 2160  
Статус: Элемент загружен из Основной памяти

Строка, изменённая ранее в КЭШ-памяти, поместилась обратно в ОП:

Моделирование работы КЭШ памяти

КЭШ память	Основная память	Сегмент:	1
-1 0 0 0 0	1890 2167 6602 7111	Строка:	3
-1 0 0 0 0	7636 8549 5893 2200	Элемент:	1
1 2160 9612 3449 7348	2160 9612 3449 7348	Найти	
-1 0 0 0 0	8754 2087 1610 5130	Элемент:	5555
2 8962 1881 6023 1181	5373 2738 9206 7665	Изменить	
-1 0 0 0 0	5081 6688 3203 5720	Время:	00:00:00.0007010
-1 0 0 0 0	2087 7455 3837 2932	Строка:	2160 9612 3449 7348
-1 0 0 0 0	6888 9098 8411 1334	Элемент:	2160
-1 0 0 0 0	9083 9361 7763 7198	Статус:	Элемент загружен из Основной памяти
-1 0 0 0 0	8023 9956 6026 1223		
	2737 7715 1035 6725		
	2801 5623 8327 7338		
	6491 1481 8747 5555		
	2718 2677 8995 5200		
	8962 1881 6023 1181		
	5711 3960 5311 3060		
	2200 3114 8263 9392		
	4323 4411 7106 5415		
	6711 8982 5927 7365		
	6653 6477 3646 6463		

Загрузка изменённого ранее элемента из Основной памяти:

Моделирование работы КЭШ памяти

КЭШ память	Основная память	Сегмент:	2
-1 0 0 0 0	1890 2167 6602 7111	Строка:	3
-1 0 0 0 0	7636 8549 5893 2200	Элемент:	4
2 6491 1481 8747 5555	2160 9612 3449 7348	Найти	
-1 0 0 0 0	8754 2087 1610 5130	Элемент:	5555
2 8962 1881 6023 1181	5373 2738 9206 7665	Изменить	
-1 0 0 0 0	5081 6688 3203 5720	Время:	00:00:00.0004220
-1 0 0 0 0	2087 7455 3837 2932	Строка:	6491 1481 8747 5555
-1 0 0 0 0	6888 9098 8411 1334	Элемент:	5555
-1 0 0 0 0	9083 9361 7763 7198	Статус:	Элемент загружен из Основной памяти
-1 0 0 0 0	8023 9956 6026 1223		
	2737 7715 1035 6725		
	2801 5623 8327 7338		
	6491 1481 8747 5555		
	2718 2677 8995 5200		
	8962 1881 6023 1181		
	5711 3960 5311 3060		
	2200 3114 8263 9392		
	4323 4411 7106 5415		
	6711 8982 5927 7365		
	6653 6477 3646 6463		

## КОД ПРОГРАММЫ

- MainForm.cs

```
using System;
using System.Diagnostics;
using System.Windows.Forms;

namespace CacheWork
{
    public partial class MainForm : Form
    {
        static int segmentsCount = 10;
        static int stringsCount = 10;
        static int elementsCount = 4;

        Controller Data = new Controller(segmentsCount, stringsCount, elementsCount);
        Stopwatch Time = new Stopwatch();

        int[] stringData;
        int I_address, J_address, K_address;
        int Values;

        public MainForm()
        {
            InitializeComponent();
            PrintMainMemory();
            PrintCache();
        }

        private void searchButton_Click(object sender, EventArgs e)
        {
            try
            {
                I_address = Convert.ToInt32(segmentTextBox.Text);
                J_address = Convert.ToInt32(stringTextBox.Text);
                K_address = Convert.ToInt32(elementTextBox.Text);

                Time.Start();
                Values = Data[I_address, J_address, K_address];
                Time.Stop();

                UpdateStatus(Values);

                PrintMainMemory();
                PrintCache();
            }
            catch
            {
                MessageBox.Show("Введены некорректные данные.", "Ошибка!",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }

        private void changeButton_Click(object sender, EventArgs e)
        {
            try
            {
                I_address = Convert.ToInt32(segmentTextBox.Text);
                J_address = Convert.ToInt32(stringTextBox.Text);
                K_address = Convert.ToInt32(elementTextBox.Text);
                Values = Convert.ToInt32(newElementTextBox.Text);

                Time.Start();
                Data[I_address, J_address, K_address] = Values;
                Time.Stop();

                UpdateStatus(Values);
            }
            catch
            {
                // This block is not visible in the image, but it follows the same pattern as the searchButton_Click method.
            }
        }
    }
}
```

```

        PrintMainMemory();
        PrintCache();
    }
    catch
    {
        MessageBox.Show("Введены некорректные данные.", "Ошибка!",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

void PrintCache()
{
    cacheTextBox.Text = "";
    for (int i = 1; i <= stringsCount; i++)
    {
        stringData = Controller.CacheController[i];
        cacheTextBox.Text += Controller.CacheController.GetTag(i);
        cacheTextBox.Text += "      ";

        for (int k = 0; k < elementsCount; k++)
        {
            cacheTextBox.Text += stringData[k] + " ";
        }

        cacheTextBox.Text += Environment.NewLine;
    }
}

void PrintMainMemory()
{
    mainMemoryTextBox.Text = "";
    for (int i = 1; i <= segmentsCount; i++)
    {
        for (int j = 1; j <= stringsCount; j++)
        {
            stringData = Controller.MainMemoryController.GetString(i, j);
            for (int k = 0; k < elementsCount; k++)
            {
                mainMemoryTextBox.Text += stringData[k] + " ";
            }
            mainMemoryTextBox.Text += Environment.NewLine;
        }
        mainMemoryTextBox.Text += Environment.NewLine;
    }
}

void UpdateStatus(int el)
{
    if (Data.FromCache)
    {
        statusTextBox.Text = " Элемент загружен из КЭШ памяти";
    }
    else
    {
        statusTextBox.Text = " Элемент загружен из Основной памяти";
    }

    elementFromTextBox.Text = el.ToString();
    timeTextBox.Text = Time.Elapsed.ToString();
    Time.Reset();

    stringFromTextBox.Text = "";
    for (int i = 1; i <= elementsCount; i++)
    {
        stringFromTextBox.Text += Controller.CacheController[J_adress, i] + " ";
    }
}
}

```

```
}
```

- Controller.cs

```
namespace CacheWork
{
    class Controller
    {
        public static MainMemory MainMemoryController;
        public static Cache CacheController;
        public bool FromCache;
        int StringsCount;

        public Controller(int segmentsCount, int stringsCount, int elementsCount)
        {
            StringsCount = stringsCount;
            MainMemoryController = new MainMemory(segmentsCount, stringsCount, elementsCount);
            CacheController = new Cache(stringsCount, elementsCount);
        }

        public int this[int i, int j, int k]
        {
            get
            {
                TagValidation(i, j);
                return CacheController[j, k];
            }
            set
            {
                TagValidation(i, j);
                CacheController[j, k] = value;
            }
        }

        // Проверяет совпадают ли требуемый и текущий теги строки
        void TagValidation(int i, int j)
        {
            FromCache = true;

            if (i != CacheController.GetTag(j))
            {
                FromCache = false;

                if (CacheController.GetTag(j) != -1)
                    MainMemoryController.SetString(CacheController.GetTag(j), j,
CacheController[j]);

                CacheController[j] = MainMemoryController.GetString(i, j);
                CacheController.SetTag(j, i);
            }
        }
    }
}
```

- MainMemory.cs

```
using System;
using System.IO;

namespace CacheWork
{
    class MainMemory
    {
        BinaryWriter BinaryWrite;
        BinaryReader BinaryRead;
```



```

int SegmentsCount,
    StringsCount,
    ElementsCount;

public MainMemory(int segmentsCount, int stringsCount, int elementsCount)
{
    Random values = new Random();
    SegmentsCount = segmentsCount;
    StringsCount = stringsCount;
    ElementsCount = elementsCount;

    using (BinaryWrite = new BinaryWriter(new FileStream("Memory.ini", FileMode.Create)))
    {
        for (int i = 0; i < SegmentsCount; i++)
        {
            BinaryWrite.Write((char)10);

            for (int j = 0; j < StringsCount; j++)
            {
                for (int k = 0; k < ElementsCount; k++)
                {
                    BinaryWrite.Write(values.Next(1000, 9999));
                    BinaryWrite.Write(' ');
                }
                BinaryWrite.Write((char)10);
            }
        }
    }

    void Positioning(int segment, int str, IDisposable WriteRead)
    {
        segment--;
        str--;

        int position = (segment + 1) + // Отступы между блоками
            (segment * (StringsCount * ((ElementsCount * 4) + 5))) + // Пропуск эл-та до
нужного сегмента
            (str * ((ElementsCount * 4) + 5)); // Пропуск эл-та до нужной строки

        // Позиция каретки с учетом размеров
        switch (WriteRead)
        {
            case BinaryWriter writer:
                writer.BaseStream.Position = position;
                break;
            case BinaryReader reader:
                reader.BaseStream.Position = position;
                break;
        }
    }

    // Получить строку в сегменте
    public int[] GetString(int segment, int str)
    {
        int[] temp = new int[ElementsCount];

        using (BinaryRead = new BinaryReader(new FileStream("Memory.ini", FileMode.Open)))
        {
            Positioning(segment, str, BinaryRead);

            for (int i = 0; i < ElementsCount; i++)
            {
                temp[i] = BinaryRead.ReadInt32();
                BinaryRead.BaseStream.Position++;
            }
        }
    }
}

```



```

        return temp;
    }

    // Записать строку temp в строку str в сегменте segment
    public void SetString(int segment, int str, int[] temp)
    {
        using (BinaryWrite = new BinaryWriter(new FileStream("Memory.ini", FileMode.Open)))
        {
            Positioning(segment, str, BinaryWrite);

            for (int i = 0; i < ElementsCount; i++)
            {
                BinaryWrite.Write(temp[i]);
                BinaryWrite.Write(' ');
            }
        }
    }
}

```

- Cache.cs

```

namespace CacheWork
{
    class Cache
    {
        (int Tag, int[] Line)[] Page;

        public Cache(int stringsCount, int elementsCount)
        {
            Page = new (int Tag, int[] Line)[stringsCount];
            for (int i = 0; i < Page.Length; i++)
            {
                Page[i].Line = new int[elementsCount];
                Page[i].Tag = -1;
            }
        }

        // Получить или изменить элемент
        public int this[int i, int j]
        {
            get => Page[i - 1].Line[j - 1];
            set => Page[i - 1].Line[j - 1] = value;
        }

        // Получить или изменить строку
        public int[] this[int i]
        {
            get => Page[i - 1].Line;
            set => Page[i - 1].Line = value;
        }

        // Получить или изменить тэг
        public int GetTag(int i) => Page[i - 1].Tag;
        public void SetTag(int i, int newTag) => Page[i - 1].Tag = newTag;
    }
}

```