МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Кафедра Автоматизированных систем управления

**ОТЧЕТ О ВЫПОЛНЕНИИ**

**Лабораторной работы № 1**

Дисциплина «*Технологии и методы программирования*»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Выполнили:** |  | **Проверил:** |  | |
| Студенты | *Бузмаков А.И.,*  *Шестаков К.Д.* | Преподаватель | *Астапчук В. А., Пустовских Д. А.* | |
| Факультет | *АВТФ* |  |  | |
| Направление (специальность) подготовки | *09.03.01 – Информатика и вычислительная техника* | Балл: \_\_\_\_\_\_\_\_\_ | |  |
| Группа | *АП-227* | Оценка *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | | |
| Шифр |  |  |  | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| подпись | | подпись | | |
| Дата сдачи: | «15» марта 2024г. | Дата защиты: | «\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. | |

Новосибирск 2024

**ВВЕДЕНИЕ**

**Тема** лабораторной работы: моделирование системы управления виртуальной памятью.

**Цель** работы: получить представление о механизме управления виртуальной памятью, закрепить навыки программирования на выбранном языке программирования с использованием динамических структур данных и прямого доступа к файлу.

**Оборудование**: персональный компьютер.

**ХОД РАБОТЫ**

**1. Описание программного продукта**

В программе реализован класс для управления виртуальной памятью. Объект этого класса моделирует массив целого типа большой размерности (>10000 элементов). Файлы, с которыми работает программа, представлены в виде документов, имеющих страницы определённого размера. Чтение файлов происходит в поточном режиме, с загрузкой в оперативную память только одной страницы, с которой работает пользователь.

Возможности программы:

* Создание файла со сгенерированными случайным образом числами (количество чисел задаёт пользователь);
* Открытие файла;
* Просмотр элементов по индексу;
* Удаление элементов;
* Изменение элементов;

**2. Описание структур работы с виртуальной памятью**

Класс VirtualMemo реализует простой виртуальный динамический массив, который сохраняет свои данные в файле на диске.

1. **VirtualMemo()** - конструктор класса. Принимает размер блока (BlockSize) и имя файла (Name). Если имя файла не указано, используется значение "Def.dat". В конструкторе инициализируются все основные переменные класса, создается битовая карта (BMap) и массив значений (Arr) заданного размера (BlockSize).
2. **CreateFile()**: Эта функция создает файл на диске и заполняет его блоками данных в соответствии с заданным пользователем количеством элементов. Она запрашивает у пользователя количество элементов, вычисляет количество необходимых страниц для хранения данных и заполняет файл нулями (либо случайными числами). В результате создается файл с заданным именем, содержащий нужное количество страниц.
3. **private int Index\_()**: Эта функция используется для ввода пользователем индекса элемента. Она запрашивает у пользователя индекс, проверяет его корректность и возвращает введенное значение. Если ввод некорректен, функция продолжает запрашивать ввод до тех пор, пока не будет получено корректное значение.
4. **ReadIn():** Получает текущий номер страницы (`NumPage`).Вызывает метод Index\_(), чтобы получить индекс нужного элемента.Вычисляет номер страницы, содержащей этот элемент, и индекс внутри страницы.Проверяет, если номер страницы больше общего количества страниц, то устанавливает NumPage в 1, иначе оставляет без изменений.Проверяет, есть ли данная страница в буфере. Если есть, копирует данные из буфера в массивы Arr и BMap и выводит сообщение о загрузке из буфера.Если страницы нет в буфере, открывает файл для чтения, перемещается к началу нужной страницы, считывает данные в массивы Arr и BMap, пропуская данные, если соответствующий флаг в BMap равен false.После чтения сохраняет страницу в буфер методом SaveInBuffer().
5. **GetBufferIndex(int pageNum):** Принимает номер страницы. Просматривает массив NumPageBuffers, чтобы найти индекс буфера, содержащего данную страницу. Возвращает индекс буфера, если страница найдена, иначе возвращает -1.
6. **SaveInBuffer():** Клонирует массивы Arr и BMap и сохраняет их в буфер ArrBuffers и BMapBuffers соответственно. Сохраняет номер страницы в массиве NumPageBuffers. Обновляет индекс буфера, чтобы определить, куда сохранить следующую страницу.
7. **ReadIndex()**: Эта функция читает значение по заданному индексу из буферного массива Arr. Она вызывает ReadIn() для загрузки нужной страницы данных в память, затем проверяет наличие значения по заданному индексу и выводит его на экран, если оно существует.
8. **EnValue()**: Эта функция используется для записи нового значения в массив Arr по текущему индексу. Она запрашивает у пользователя новое значение, записывает его в массив Arr и устанавливает соответствующий бит в битовой карте BMap. После этого вызывается функция WriteFile(), чтобы сохранить изменения на диск.
9. **Delete()**: Эта функция удаляет значение из массива Arr по текущему индексу. Она устанавливает значение на данном индексе в ноль и устанавливает соответствующий бит в битовой карте BMap в false. Затем вызывается функция WriteFile(), чтобы сохранить изменения на диск.
10. **EnDel()**: Эта функция используется для замены значения по текущему индексу. Она сначала вызывает ReadIn(), чтобы загрузить данные с диска в память, затем проверяет наличие значения по текущему индексу. Если значение существует, функция предлагает пользователю удалить его и записать новое значение или оставить текущее значение без изменений.
11. **WriteFile()**: Эта функция записывает текущее состояние буферного массива Arr и битовой карты BMap в файл на диске. Она открывает файл для записи, перемещается к соответствующей странице данных и записывает данные из массивов Arr и BMap в файл.

**3. Тестирование программы**

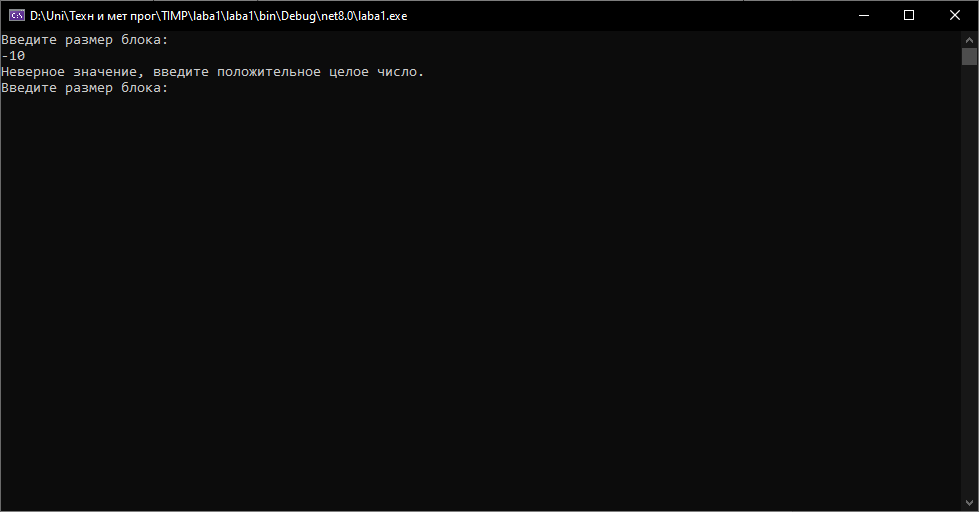


Рис. 1 - Ввод некорректного значения размера блока (страницы)

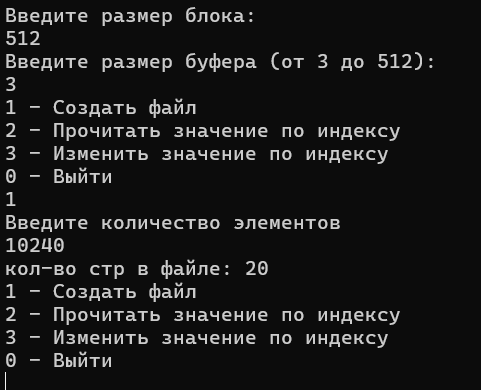


Рис. 2 - Создание файла

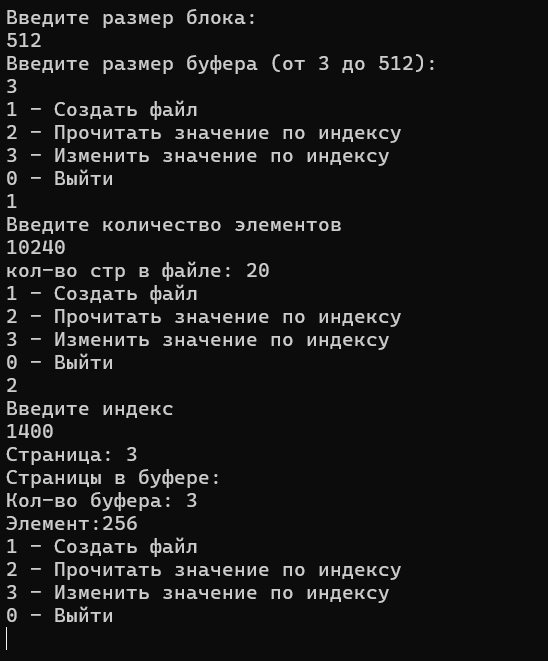


Рис. 3 - Вывод элемента

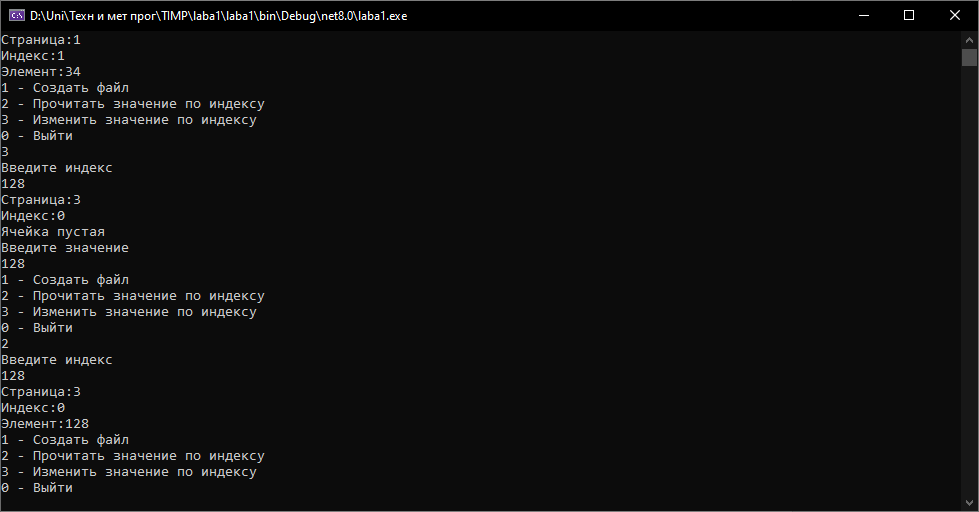


Рис. 4 - Запись значения

В ходе работы программы так же проводился мониторинг файла дампа, файл после создания имел следующий вид (Рис. 5):

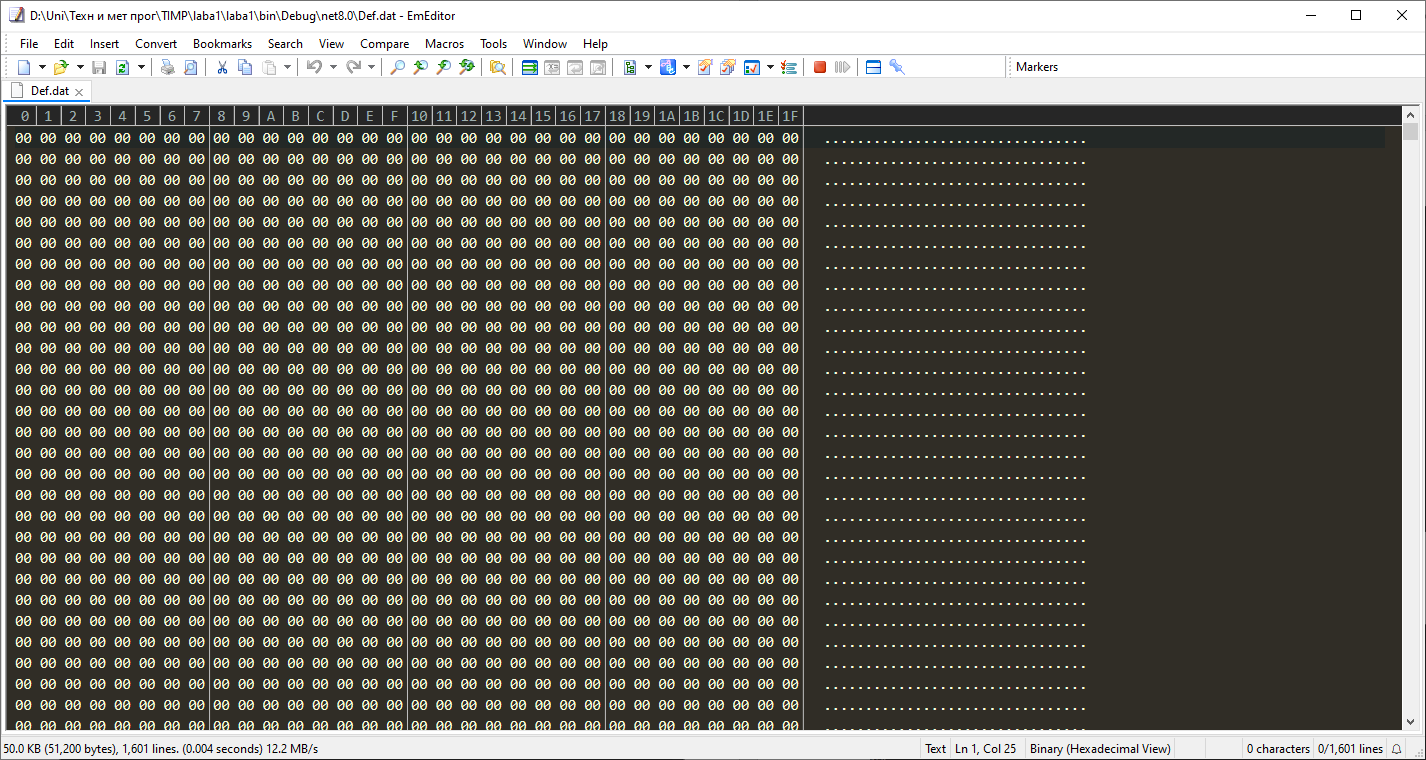


Рис. 5 - Двоичный вид файла до изменений

Введем в ячейку с индексом 0 значение 32 (Рис. 6):

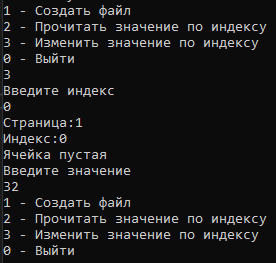


Рис. 6 - Добавление значения в файл

После добавления элемента содержимое файла приняло следующий вид:

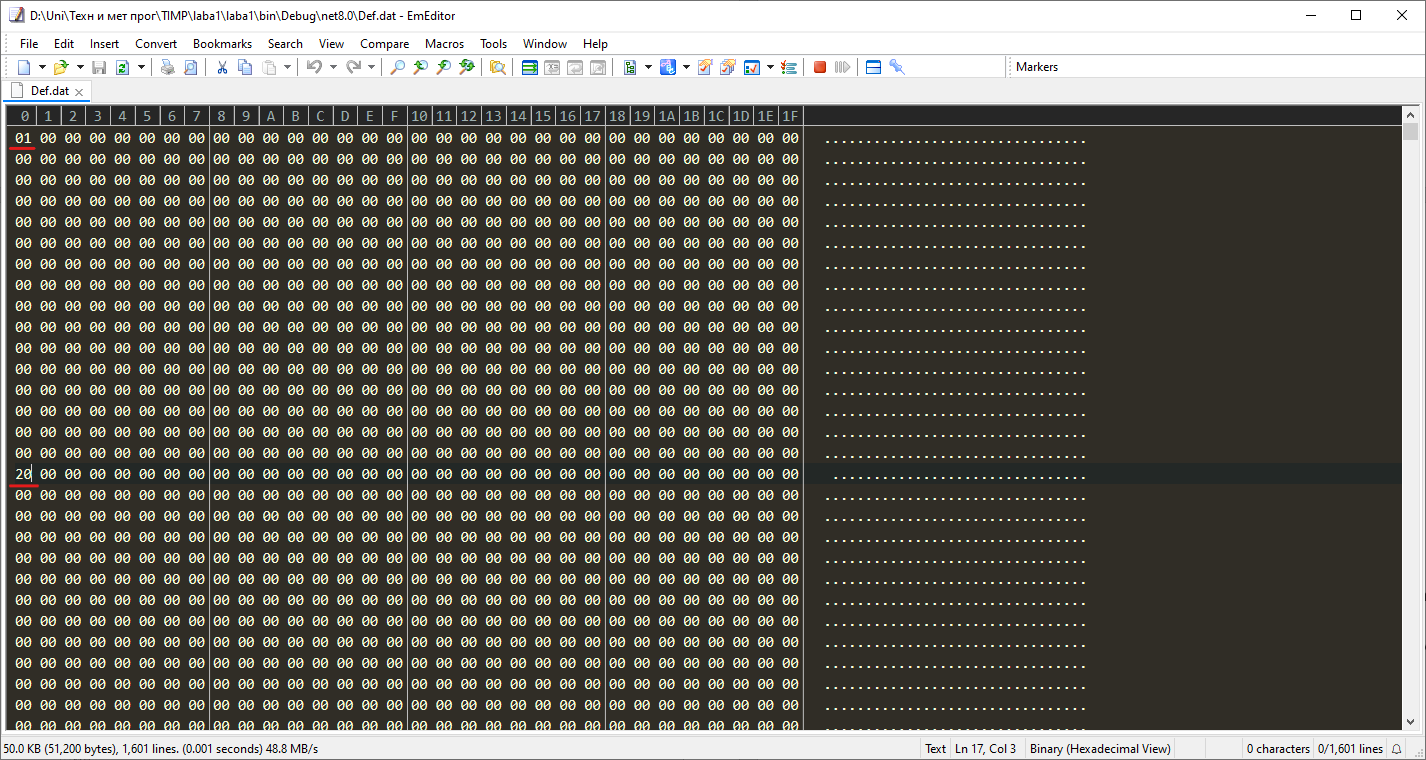


Рис. 7 - Двоичный файл после изменения

Как видно на Рис. 7, в файл были записаны как значение 32, так и флаг, показывающий что в ячейке с индексом 0 записано число.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы были получены представления о механизме управления виртуальной памятью, закреплены навыки программирования на выбранном языке C# с использованием динамических структур данных и прямого доступа к файлу.

**ПРИЛОЖЕНИЕ**

|  |
| --- |
| VirtualMemo.cs |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Text;  namespace laba1  {  class VirtualMemo  {  string Name;  int BlockSize;  int[] Arr; // Массив значений  int[] ArrPast;  bool[] BMap; // Битовая карта  bool[] BMapPast;  int Index; // Индекс на странице  int NumPagePast = -1;  int NumPage = -1; // Номер страницы  int Pages;  private int BufferSize;  private int[][] ArrBuffers;  private bool[][] BMapBuffers;  private int[] NumPageBuffers;  private int bufferIndex = 0; // Индекс текущего буфера в кольцевом буфере  // Конструктор  public VirtualMemo(int BlockSize, int BufferSize, string Name = "Def.dat")  {  this.BlockSize = BlockSize;  this.Name = Name;  this.BufferSize = BufferSize;  ArrBuffers = new int[BufferSize][];  BMapBuffers = new bool[BufferSize][];  NumPageBuffers = new int[BufferSize];  BMap = new bool[BlockSize];  BMapPast = new bool[BlockSize];  Arr = new int[BlockSize];  ArrPast = new int[BlockSize];  }  // Создание файла  public void CreateFile()  {  int el;  while (true)  {  Console.WriteLine("Введите количество элементов");  string b = Console.ReadLine();  bool success = int.TryParse(b, out el);  if (success)  {  if (el >= 1) break;  else Console.WriteLine("Неверный ввод"); continue;  }  else  {  Console.WriteLine("Неверный ввод"); continue;  }  }  using (FileStream fs = new FileStream(Name, FileMode.Create))  {  using (BinaryWriter bw = new BinaryWriter(fs, Encoding.UTF8))  {  int pages = el / BlockSize;  int lastp = el - pages \* BlockSize;  for (int i = 0; i < pages; i++)  {  bw.Seek((sizeof(bool) \* BMap.Length + sizeof(int) \* Arr.Length) \* (i), 0);  Random rnd = new Random();  for (int j = 0; j < BlockSize; j++)  {  int b = rnd.Next(0, 2);  if (b == 0)  {  BMap[j] = false;  bw.Write(false);  }  else  {  BMap[j] = true;  bw.Write(true);  }  }  for (int j = 0; j < BlockSize; j++)  {  if (BMap[j] == true)  bw.Write(rnd.Next(1, 255));  else bw.Write(0);  }  }  if (lastp != 0)  {  pages = pages + 1;  Random rnd = new Random();  for (int j = 0; j < BlockSize; j++)  {  if (j < lastp)  {  int b = rnd.Next(0, 2);  if (b == 0)  {  BMap[j] = false;  bw.Write(false);  }  else  {  BMap[j] = true;  bw.Write(true);  }  }  else bw.Write(false);  }  for (int j = 0; j < BlockSize; j++)  {  if (BMap[j] == true)  bw.Write(rnd.Next(1, 255));  else bw.Write(0);  }  }  Pages = pages;  Console.WriteLine("кол-во стр в файле: " + Pages);  }  }  }  // Ввод индекса  private int Index\_()  {  int index;  while (true)  {  Console.WriteLine("Введите индекс");  string b = Console.ReadLine();  bool success = int.TryParse(b, out index);  if (success)  {  if (index >= 0) return index;  else Console.WriteLine("Неверный ввод"); continue;  }  else  {  Console.WriteLine("Неверный ввод"); continue;  }  }  }  // Чтение файла  public void ReadIn()  {  int tempPage = NumPage;  int index = Index\_(); // Предположим, что Index\_() возвращает правильный индекс  int pageNum = (index / BlockSize) + 1;  Index = index % BlockSize;  if (pageNum > Pages)  NumPage = 1;  else  NumPage = pageNum;  int bufferIndex = GetBufferIndex(NumPage);  if (bufferIndex != -1)  {  // Скопировать данные из буфера  Array.Copy(ArrBuffers[bufferIndex], Arr, ArrBuffers[bufferIndex].Length);  Array.Copy(BMapBuffers[bufferIndex], BMap, BMapBuffers[bufferIndex].Length);  Console.WriteLine($"Страница №{NumPage} загружена из буфера.");  }  else  {  using (BinaryReader reader = new BinaryReader(File.Open(Name, FileMode.Open)))  {  // Переход к началу нужной страницы в файле  reader.BaseStream.Seek((NumPage - 1) \* BlockSize, SeekOrigin.Begin);  Console.WriteLine("Страница: " + NumPage);  Console.Write("Страницы в буфере: ");  foreach (var page in NumPageBuffers)  {  if (page != 0)  {  Console.Write(page + " ");  }  }  Console.WriteLine("");  Console.WriteLine("Кол-во буфера: " + NumPageBuffers.Length);  // Чтение данных страницы  for (int i = 0; i < BlockSize; i++)  BMap[i] = reader.ReadBoolean();  for (int i = 0; i < BlockSize; i++)  {  if (BMap[i])  Arr[i] = reader.ReadInt32();  else  reader.ReadInt32(); // Пропустить данные, если флаг равен false  }  }  // Сохранить страницу в буфер, только если она не была прочитана из буфера  SaveInBuffer();  }  NumPagePast = tempPage;  }  private int GetBufferIndex(int pageNum)  {  for (int i = 0; i < BufferSize; i++)  {  if (NumPageBuffers[i] == pageNum)  return i;  }  return -1;  }  private void SaveInBuffer()  {  // Сохранить текущую страницу в буфер  ArrBuffers[bufferIndex] = (int[])Arr.Clone();  BMapBuffers[bufferIndex] = (bool[])BMap.Clone();  NumPageBuffers[bufferIndex] = NumPage;  // Обновить индекс буфера  bufferIndex = (bufferIndex + 1) % BufferSize;  }  // Чтение по индексу  public void ReadIndex()  {  ReadIn();  if (BMap[Index] == true)  {  Console.WriteLine("Элемент:" + Arr[Index]);  }  else  {  Console.WriteLine("Ячейка пуста");  }  }  // Запись значения  public void EnValue()  {  int el;  while (true)  {  Console.WriteLine("Введите значение");  string b = Console.ReadLine();  bool success = int.TryParse(b, out el);  if (success == false)  {  Console.WriteLine("Неверный ввод"); continue;  }  else  {  break;  }  }  Arr[Index] = el;  BMap[Index] = true;  WriteFile();  }  // Удаление значения  public void Delete()  {  Arr[Index] = 0;  BMap[Index] = false;  WriteFile();  }  // Замена значения  public void EnDel()  {  ReadIn();  if (BMap[Index] == true)  {  Console.WriteLine("Удалить старое значение?: \n 1 - да \n 2 - нет ");  int a;  a = Convert.ToInt32(Console.ReadLine());  switch (a)  {  case 1:  {  Delete();  EnValue();  break;  }  case 2: Console.WriteLine("Значение не изменено"); break;  default: Console.WriteLine("Ошибка ввода"); break;  }  }  else  {  Console.WriteLine("Ячейка пустая");  EnValue();  }  }  // Запись файла  private void WriteFile()  {  using (BinaryWriter writer = new BinaryWriter(File.Open(Name, FileMode.OpenOrCreate)))  {  writer.Seek((sizeof(bool) \* BMap.Length + sizeof(int) \* Arr.Length) \* (NumPage - 1), 0);  for (int i = 0; i < BlockSize; i++)  {  writer.Write(BMap[i]);  }  for (int i = 0; i < BlockSize; i++)  {  writer.Write(Arr[i]);  }  }  }  }  } |
| Program.cs |
| namespace laba1  {  class Program  {  static void Main(string[] args)  {  int blockSize = 0;  int bufferSize = 0;  bool validInput = false;  bool validInput2 = false;  while (!validInput)  {  Console.WriteLine("Введите размер блока:");  string input = Console.ReadLine();  if (int.TryParse(input, out blockSize) && blockSize > 0)  {  validInput = true;  }  else  {  Console.WriteLine("Неверное значение, введите положительное целое число.");  }  }  while (!validInput2)  {  Console.WriteLine("Введите размер буфера (от 3 до 512):");  string input = Console.ReadLine();  if (int.TryParse(input, out bufferSize) && bufferSize >= 3 && bufferSize <= 512)  {  validInput2 = true;  }  else  {  Console.WriteLine("Неверное значение, введите положительное целое число от 3 до 512.");  }  }  VirtualMemo virtualMemo = new VirtualMemo(blockSize, bufferSize);  while (true)  {  Console.WriteLine("1 - Создать файл");  Console.WriteLine("2 - Прочитать значение по индексу");  Console.WriteLine("3 - Изменить значение по индексу");  Console.WriteLine("0 - Выйти");  var IsValid = false;  int choice = -1;  while(IsValid != true)  {  IsValid = Int32.TryParse(Console.ReadLine(), out choice);  if(IsValid == false)  {  Console.WriteLine("Неверный ввод, попробуйте еще раз.");  }  }  switch (choice)  {  case 0:  return;  case 1:  virtualMemo.CreateFile();  break;  case 2:  virtualMemo.ReadIndex();  break;  case 3:  virtualMemo.EnDel();  break;  default:  Console.WriteLine("Неверный выбор.");  break;  }  }  }  }  } |