МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИСТ

Кафедра «Информационные системы»

Дисциплина «Объектно-ориентированное программирование»

**КУРСОВАЯ РАБОТА**

Тема «Разработка программы «Визуализация задачи о рюкзаке «Жадный алгоритм»

Выполнил студент \_ /Гуторов И.А./

подпись инициалы, фамилия

Курс первый Группа ПИбд-13

Направление/специальность 09.03.04 «Программная инженерия» (профиль «Искусственный интеллект и предиктивная аналитика»)

Руководитель доцент Эгов Е.Н.

должность, ученая степень, ученое звание фамилия, имя, отчество

Дата сдачи:

« » 2024 г.

Дата защиты:

« » 2024 г. Оценка:

Ульяновск 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ИСТ

Кафедра «Информационные системы»

Дисциплина «Объектно-ориентированное программирование»

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студенту ПИбд-13 Гуторов И.А.

группа фамилия, инициалы

Тема работы «Разработка программы «Визуализация задачи о рюкзаке «Жадный алгоритм»

Срок сдачи законченной работы « » 2024 г.

**Исходные данные к работе:** описание задания по теме, утвержденной распоряжением деканата ФИСТ

**Рекомендуемая литература:** курс лекций по дисциплине «Объектно- ориентированное программирование», методические указания к лабораторным работам по дисциплине «Объектно-ориентированное программирование», интернет- источники.

**Содержание пояснительной записки** (перечень подлежащих разработке вопросов)

Первая глава. Описание алгоритма решения задачи/алгоритма/АТД/операции.

Вторая глава. Представление руководства пользователя для разработанного проекта. Третья глава. Представление руководства программиста для разработанного проекта. **Перечень графического материала** (с точным указанием обязательных чертежей) Скриншоты разработанного программного продукта.

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель | доцент | \_ | /Эгов Е.Н./ |
|  | должность | подпись | инициалы, фамилия |

« » \_ 2024 г.

Студент /Гуторов И.А./

подпись инициалы, фамилия

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ**

**руководителя на курсовую работу**

студента Гуторова Ивана Александровича

фамилия, имя и отчество

Факультет ИСТ группа ПИбд-13 курс первый Дисциплина «Объектно-ориентированное программирование» Тема работы «Разработка программы «Визуализация задачи о рюкзаке «Жадный алгоритм»

Руководитель доцент /Эгов Е.Н./

должность, учёная степень, ученое звание подпись инициалы, фамилия

« » \_ 2024 г.

# ПЕРВАЯ ГЛАВА. ОПИСАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ О РЮКЗАКЕ ЖАДНЫМ АЛГОРИТМОМ

История алгоритма

Жадный алгоритм для задачи о дробном рюкзаке впервые появился в литературе в 1957 году благодаря работе Динкельбаха и Шроера, однако первым, кто предложил жадный алгоритм как решение этой задачи, был Джордж Данциг. В своей статье они предложили алгоритм, который основывался на ранжировании предметов по убыванию их удельной стоимости (стоимость на единицу веса). Идея заключалась в том, чтобы добавлять в рюкзак предметы с наибольшей удельной стоимостью до тех пор, пока это возможно.

Формулировка

Имеется рюкзак, который может выдержать вес . Даны предметов, каждый из которых имеет вес и ценность . Необходимо выбрать подмножество предметов так, чтобы суммарный вес выбранных предметов не превышал , а их суммарная ценность была максимальной.

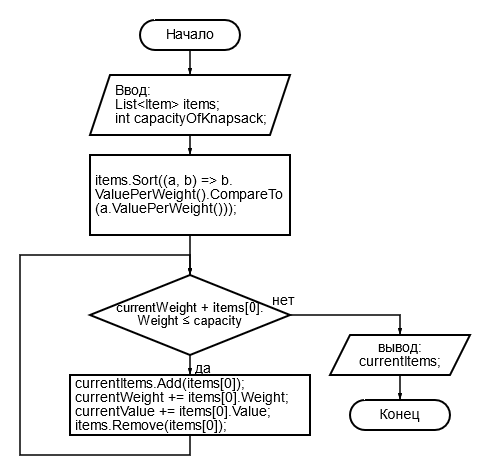
Алгоритм решения

Жадный алгоритм основывается на выборе локально оптимальных решений на каждом шаге в надежде, что это приведет к глобально оптимальному решению.

Для задачи о рюкзаке жадный алгоритм можно описать следующим образом:

* Сортировка предметов: Предметы сортируются по убыванию удельной ценности, где удельная ценность предмета определяется как отношение его ценности к весу .
* Выбор предметов: Предметы добавляются в рюкзак в порядке убывания их удельной ценности до тех пор, пока не останется ни одного предмета, добавление следующего предмета не приведет к превышению допустимого веса .

Блок-схема



# ГЛАВА ВТОРАЯ. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

После запуска появляется главная форма (рис. 1). Для того чтобы задать начальные параметры, нужно нажать кнопку «Задать параметры».

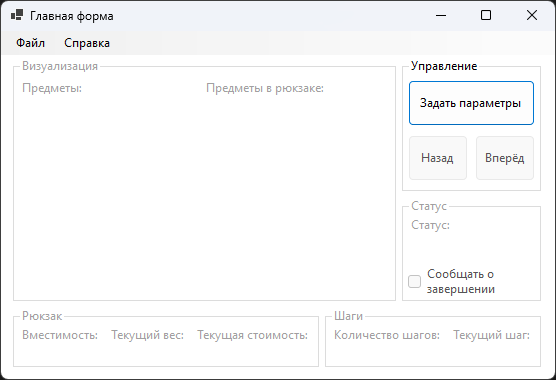


Рисунок 1. Главная форма после запуска

После нажатия на кнопку появится форма ввода (рис. 2).

Есть 3 поля ввода:

1. Вес – вес добавляемого в список предмета, принимает значения от 1 до 100.
2. Стоимость – стоимость добавляемого в список предмета, принимает значения от 1 до 100.
3. Вместимость рюкзака ­– вместимость рюкзака, принимает значения от 1 до 100.

Есть 3 кнопки:

1. Кнопка «Добавить предмет» – отвечает за добавления предмета в список начальных предметов.
2. Кнопка «Сбросить» – сбрасывает выставленные пользователем значения веса, стоимости и вместимости рюкзака, а также очищает список начальных предметов.
3. Кнопка «Принять» – отвечает за передачу введённых параметров на главную форму.

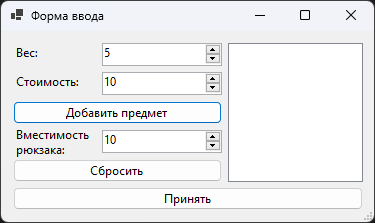


Рисунок 2. Форма ввода

С помощью кнопок можно задать список предметов и вместимость рюкзака (рис. 3).

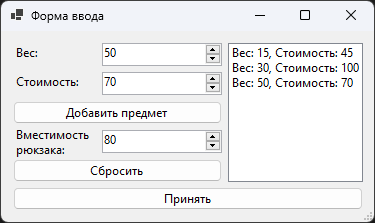
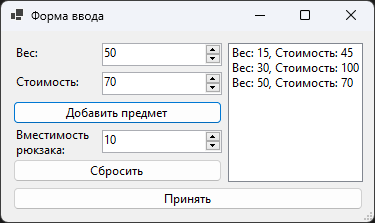
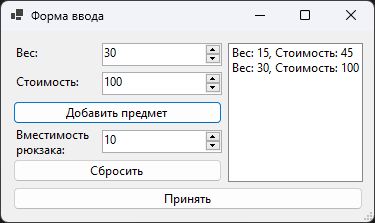
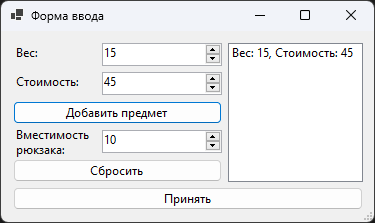


Рисунок 3. Поэтапное заполнение предметами и установка вместимости рюкзака

После выставления свои параметров нажимаем кнопку «Принять» и переходим на главную форму (рис. 4). На ней в виде прямоугольников отображены предметы, в которых написаны их свойства (В – вес, С – стоимость, С/В – удельная стоимость). Есть панель «Рюкзак» с информацией о рюкзаке на текущем шаге и в целом (вместимость, текущий вес, текущая стоимость), панель «Шаги» с информацией об общем количестве шагов и текущем шаге, панель «Статус», помогающая понять, какой этап алгоритма отображён на текущем шаге, панель «Управление» с кнопками для задачи параметров и обхода по шагам и панель «Визуализация» с, собственно, визуалиацией.

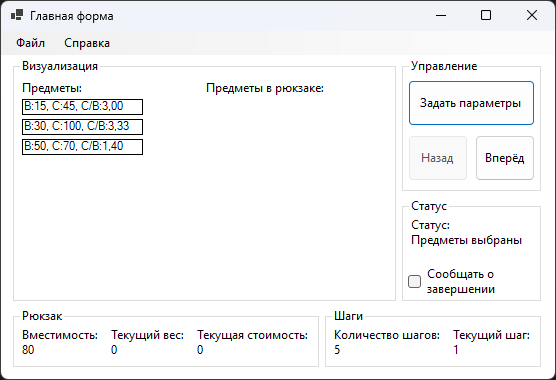


Рисунок 4. Главная форма после выставления параметров

Далее, нажимая кнопки «Вперёд» и «Назад», ходим по шагам (рис. 5).

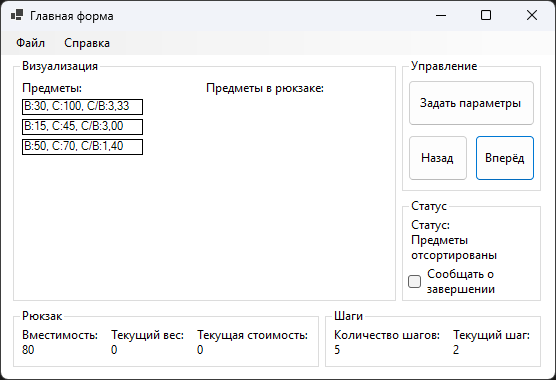
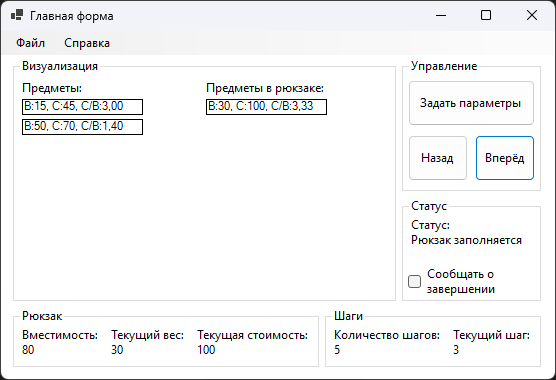
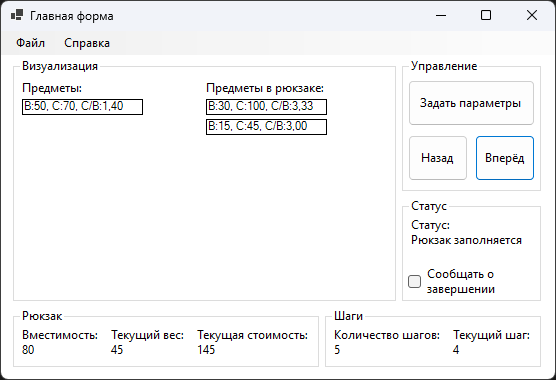
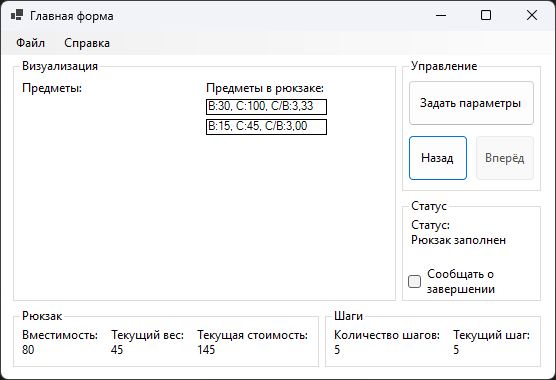


Рисунок 5. Пошаговое выполнение алгоритма

Также, есть функция сообщения о завершении (рис. 6).

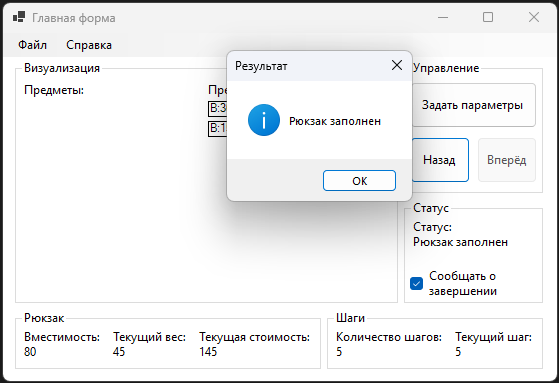


Рисунок 6. Поставлена галочка «Сообщать о завершении»

Для просмотра информации об алгоритме, есть форма информации, открывающаяся по нажатию на пункт «Справка» (рис. 7).

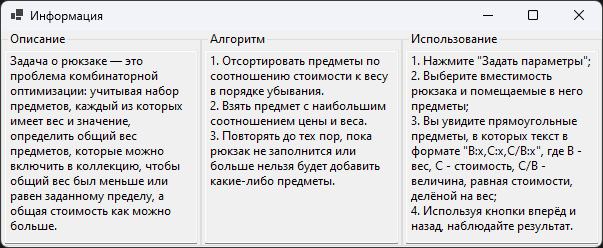


Рисунок 7. Форма информации

Есть возможность сохранения в бинарный файл (рис. 8).

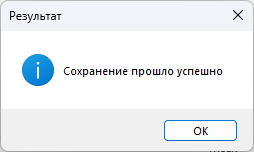
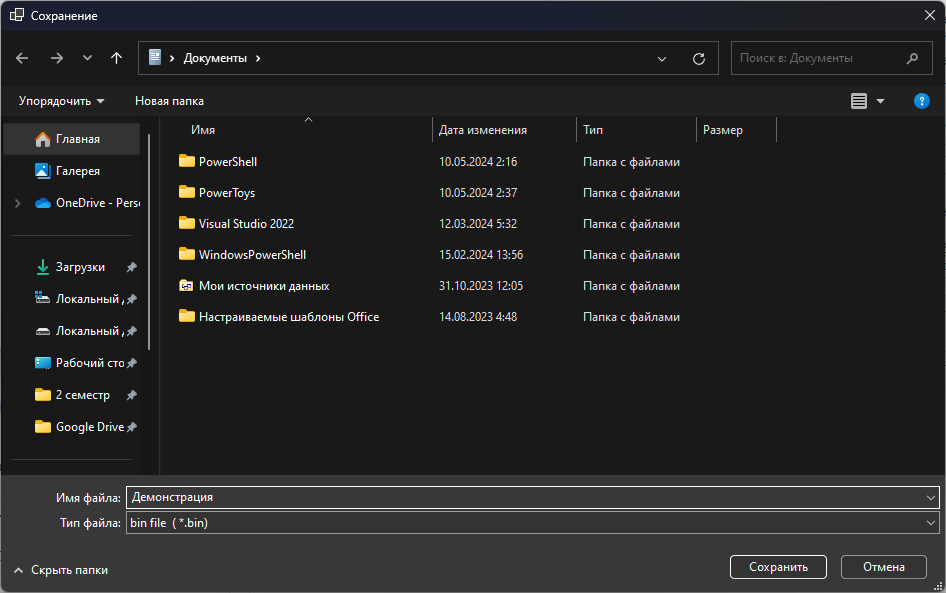
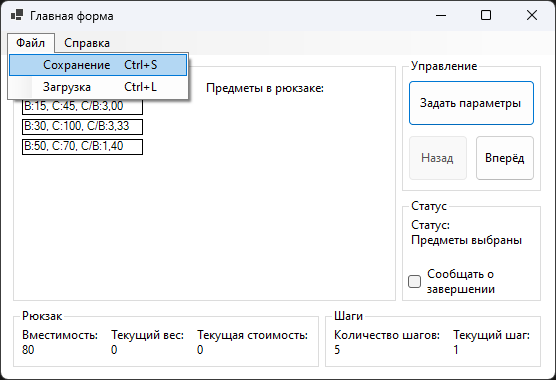


Рисунок 8. Сохранение в бинарный файл

Есть возможность загрузки из бинарного файла (рис. 9).

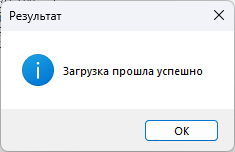
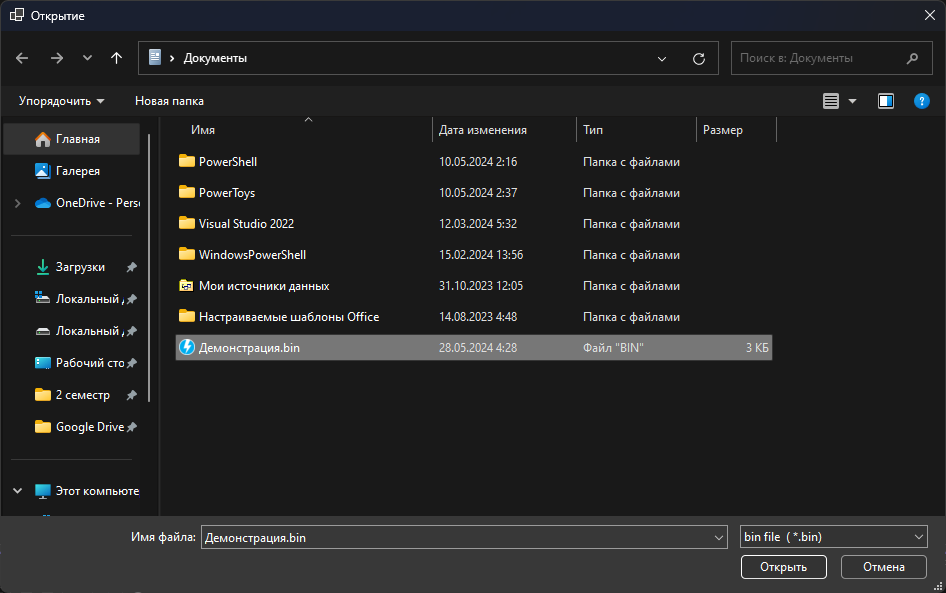
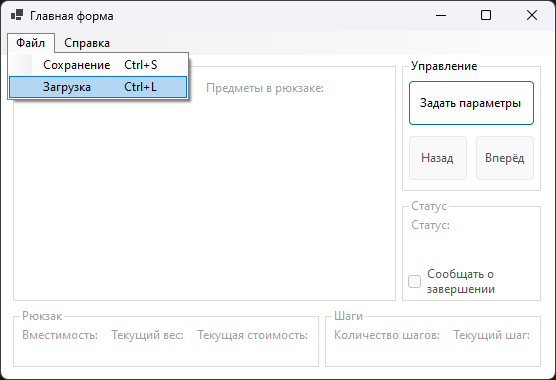


Рисунок 9. Загрузка из бинарного файла

# ГЛАВА ТРЕТЬЯ. РУКОВОДСТВО ПРОГРАММИСТА

Написанные классы:

1. KnapsackSolver – класс-реализатор. Решает задачу о рюкзаке жадным алгоритмом.

Методы:

* + public KnapsackState Step() – класс делает шаг по алгоритму и возвращает состояние на сделанном шаге.
  + public KnapsackState GetState() – метод возвращает текущее состояние.

1. Item – класс-предмет. Хранит данные о предмете.

Свойства:

* public int Weight { get; } – вес предмета.
* public int Value { get; } – стоимость предмета.

Методы:

* public double ValuePerWeight() – возвращает удельную стоимость предмета (стоимость, делёную на вес).

1. KnapsackParameters – класс-параметр. Хранит введённые в форме ввода параметры.

Свойства:

* public List<Item> Items { get; } – предметы.
* public int Capacity { get; } – вместимость рюкзака.

1. KnapsackState – класс-состояние. Хранит в себе всю информацию о конкретном шаге алгоритма.

Свойства:

* public List<Item> Items { get; } – предметы, оставшиеся для добавления.
* public int Capacity { get; } – вместимость рюкзака.
* public List<Item> CurrentItems { get; } – предметы, добавленные в рюкзак.
* public int TotalWeight { get; } – текущий вес рюкзака.
* public int TotalValue { get; } – текущая стоимость рюкзака.
* public int NumStep { get; } – номер шага.

1. KnapsackStateStorage – класс-хранилище. Хранит все состояния.

Свойства:

* public int CountOfStates => states.Count; – количество состояний.

Методы:

* public void AddState(KnapsackState state) – добавление нового состояния.
* public KnapsackState GetNextState() – получение следующего состояния и переход на него внутри класса.
* public KnapsackState GetPreviousState() – получение предыдущего состояния и переход на него внутри класса.
* public KnapsackState? Reset() – возврат на начало набора состояний.
* public void SaveToFile(string filePath) – сохранение в бинарный файл.
* public void LoadFromFile(string filePath) – загрузка из бинарного файла.

1. KnapsackManager – класс-управленец. Хранит в себе реализатор и хранилище, заполняемое с помощью реализатора.

Свойства:

* public KnapsackStateStorage Storage { get; } – хранилище состояний.

Методы:

* public void Run() – запуск алгоритма и записи состояний в хранилище.

1. MainForm – главная форма. На ней происходит визуализация.
2. InputForm – форма ввода, на которой происходит ввод параметров.

Свойства:

* public KnapsackParameters? Parameters { get; private set; } – параметры, передаваемые на главную форму.
* public event EventHandler? ParametersSubmitted; – делегат, через который параметры передаются на форму.

1. InformationForm – форма информации, отображающая только информацию о программе.

# ЗАКЛЮЧЕНИЕ

Завершено написание оконного приложения на языке C#. В процессе создания программы были улучшены навыки программирования, расширены знания об алгоритмах, улучшены навыки трассировки кода, освоены некоторые шаблоны проектирования программ.

Это приложение будет улучшаться и дальше, поэтому обучение на написании этой конкретной программы ещё не окончено.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Pro Git. 2nd Edition [Электронный ресурс] / Режим доступа: [https://git-](https://git-scm.com/book/ru/v2) [scm.com/book/ru/v2](https://git-scm.com/book/ru/v2).
2. METANIT.COM. Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com/sharp/>. – Загл. с экрана.
3. ProfessorWeb. .Net & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru/>. – Загл. с экрана.
4. Tiberiu Covaci, Rod Stephens, Vincent Varallo, Gerry O'Brien. MCSD Certification Toolkit (Exam 70-483) // Published by John Wiley & Sons, Inc. – 2013. – 656p.
5. MCTS Self-Paced Training Kit (Exam 70-536): Microsoft .NET Framework–Application Development Foundation, Second Edition eBook // Published by Microsoft Press. – 2009. – 829 p.
6. Задача о рюкзаке — Викиконспекты [Электронный ресурс] / Режим доступа: <https://neerc.ifmo.ru/wiki/index.php?title=%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%80%D1%8E%D0%BA%D0%B7%D0%B0%D0%BA%D0%B5>. – Загл. с экрана.
7. Задача о рюкзаке - Основы алгоритмов [Электронный ресурс] / Режим доступа: https://education.yandex.ru/handbook/algorithms/article/zadacha-o-ryukzake. – Загл. с экрана.
8. Теоретический материал: Задача о рюкзаке [Электронный ресурс] / Режим доступа: <https://informatics.msk.ru/mod/book/view.php?id=815&chapterid=60>. –Загл. с экрана.
9. Документация по .NET | Microsoft Learn [Электронный ресурс] / Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/>. – Загл. с экрана.
10. Andrey on .NET | Про C#, .NET, ASP.NET, Core, MVC, Azure, EF, IoC и другие умные слова [Электронный ресурс] / <https://andrey.moveax.ru/>. – Загл. с экрана.

# ПРИЛОЖЕНИЕ 1. ЛИСТИНГ КОДА

|  |
| --- |
| namespace CourseWork;  /// <summary>  /// Класс-реализатор  /// </summary>  public class KnapsackSolver  {  private List<Item> items;  private int capacity;  private List<Item> currentItems;  private int currentWeight;  private int currentValue;  private int step;  public KnapsackSolver(KnapsackParameters parameters)  {  this.items = new List<Item>(parameters.Items);  this.capacity = parameters.Capacity;  this.currentItems = new List<Item>();  this.currentWeight = 0;  this.currentValue = 0;  this.step = 0;  }  public KnapsackState Step()  {  if (step == 0)  {  step++;  return GetState();  }  else if (step == 1)  {  items.Sort((a, b) => b.ValuePerWeight().CompareTo(a.ValuePerWeight()));  step++;  return GetState();  }  else if (currentWeight + items[0].Weight <= capacity)  {  currentItems.Add(items[0]);  currentWeight += items[0].Weight;  currentValue += items[0].Value;  items.Remove(items[0]);  step++;  return GetState();  }  else if (!FittingObjects() || currentWeight == capacity || items.Count == 0)  {  items.Clear();  step++;  return GetState();  }  else if (FittingObjects())  {  items.Remove(items[0]);  step++;  return GetState();  }  return GetState();  }  public KnapsackState GetState()  {  return new KnapsackState(items, capacity, currentItems, currentWeight, currentValue, step);  }  private bool FittingObjects()  {  foreach (Item item in items)  {  if (currentWeight + item.Weight <= capacity)  return true;  }  return false;  }  }  [Serializable]  public class Item  {  public int Weight { get; }  public int Value { get; }  public Item(int weight, int value)  {  Weight = weight;  Value = value;  }  public double ValuePerWeight()  {  return (double)Value / Weight;  }  } |

|  |
| --- |
| namespace CourseWork;  /// <summary>  /// Класс-параметр  /// </summary>  public class KnapsackParameters  {  public List<Item> Items { get; }  public int Capacity { get; }  public KnapsackParameters(List<Item> items, int capacity)  {  Items = items;  Capacity = capacity;  }  } |

|  |
| --- |
| namespace CourseWork;  /// <summary>  /// Класс-состояние  /// </summary>  [Serializable]  public class KnapsackState  {  public List<Item> Items { get; }  public int Capacity { get; }  public List<Item> CurrentItems { get; }  public int TotalWeight { get; }  public int TotalValue { get; }  public int NumStep { get; }  public KnapsackState(List<Item> items, int capacity, List<Item> currentItems, int totalWeight, int totalValue, int numStep)  {  Items = new List<Item>(items);  Capacity = capacity;  CurrentItems = new List<Item>(currentItems);  TotalWeight = totalWeight;  TotalValue = totalValue;  NumStep = numStep;  }  } |

|  |
| --- |
| using System.Runtime.Serialization.Formatters.Binary;  namespace CourseWork;  /// <summary>  /// Класс-хранилище  /// </summary>  [Serializable]  public class KnapsackStateStorage  {  private List<KnapsackState> states;  private int currentIndex;  public int CountOfStates => states.Count;  public KnapsackStateStorage()  {  states = new List<KnapsackState>();  currentIndex = 0;  }  public void AddState(KnapsackState state)  {  states.Add(state);  }  public KnapsackState GetNextState()  {  if (currentIndex < CountOfStates - 1)  {  return states[++currentIndex];  }  return states[currentIndex];  }  public KnapsackState GetPreviousState()  {  if (currentIndex > 0)  {  return states[--currentIndex];  }  return states[currentIndex];  }  public KnapsackState? Reset()  {  if (states.Count > 0)  {  currentIndex = 0;  return states[currentIndex];  }  return null;  }  public void SaveToFile(string filePath)  {  using (FileStream fs = new(filePath, FileMode.Create))  {  #pragma warning disable SYSLIB0011  BinaryFormatter formatter = new();  formatter.Serialize(fs, this);  #pragma warning restore SYSLIB0011  }  }  public void LoadFromFile(string filePath)  {  using (FileStream fs = new FileStream(filePath, FileMode.Open))  {  #pragma warning disable SYSLIB0011  BinaryFormatter formatter = new();  states = ((KnapsackStateStorage)formatter.Deserialize(fs)).states;  #pragma warning restore SYSLIB0011  }  }  } |

|  |
| --- |
| namespace CourseWork;  /// <summary>  /// Класс-визуализатор  /// </summary>  public class KnapsackVisualizer  {  private PictureBox pictureBox;  public KnapsackVisualizer(PictureBox pictureBox)  {  this.pictureBox = pictureBox;  }  public void Visualize(KnapsackState state)  {  Bitmap bitmap = new(pictureBox.Width, pictureBox.Height);  if (state != null)  {  using (Graphics g = Graphics.FromImage(bitmap))  {  g.Clear(Color.White);  DrawState(g, state);  }  }  pictureBox.Image = bitmap;  }  private void DrawState(Graphics g, KnapsackState state)  {  int startXItems = 6;  int startXCurrentItems = 190;  int startY = 22;  int itemWidth = 120;  int itemHeight = 15;  int margin = 5;  int countItems = 0;  int countCurrentItems = 0;  foreach (Item item in state.Items)  {  Rectangle rectangle = new(startXItems, startY + countItems \* (itemHeight + margin), itemWidth, itemHeight);  g.DrawRectangle(Pens.Black, rectangle);  g.DrawString($"В:{item.Weight}, С:{item.Value}, С/В:{item.ValuePerWeight():F2}",  SystemFonts.DefaultFont, Brushes.Black, startXItems, startY + countItems \* (itemHeight + margin));  countItems++;  }  foreach (Item item in state.CurrentItems)  {  Rectangle rectangle = new(startXCurrentItems, startY + countCurrentItems \* (itemHeight + margin), itemWidth, itemHeight);  g.DrawRectangle(Pens.Black, rectangle);  g.DrawString($"В:{item.Weight}, С:{item.Value}, С/В:{item.ValuePerWeight():F2}",  SystemFonts.DefaultFont, Brushes.Black, startXCurrentItems, startY + countCurrentItems \* (itemHeight + margin));  countCurrentItems++;  }  }  } |

|  |
| --- |
| namespace CourseWork;  /// <summary>  /// Класс-управленец  /// </summary>  public class KnapsackManager  {  private KnapsackSolver solver;  public KnapsackStateStorage Storage { get; }  public KnapsackManager(KnapsackParameters parameters)  {  this.solver = new KnapsackSolver(parameters);  this.Storage = new KnapsackStateStorage();  }  public void Run()  {  Storage.Reset();  while (solver.GetState().Items.Count != 0)  {  Storage.AddState(solver.Step());  }  }  } |

|  |
| --- |
| namespace CourseWork;  /// <summary>  /// Главная форма  /// </summary>  public partial class MainForm : Form  {  private KnapsackManager? manager;  private KnapsackVisualizer visualizer;  public MainForm()  {  InitializeComponent();  visualizer = new KnapsackVisualizer(pictureBox);  }  private void ButtonRun\_Click(object sender, EventArgs e)  {  ShowInputForm();  }  private void ShowInputForm()  {  InputForm inputForm = new();  inputForm.ParametersSubmitted += InputForm\_ParametersSubmitted;  inputForm.ShowDialog();  }  private void InputForm\_ParametersSubmitted(object? sender, EventArgs e)  {  var inputForm = sender as InputForm;  if (inputForm != null && inputForm.Parameters != null)  {  manager = new KnapsackManager(inputForm.Parameters);  manager.Run();  KnapsackState? state = manager.Storage.Reset();  if (state != null)  {  InitializeStateShowing(state);  }  }  }  private void InitializeStateShowing(KnapsackState state)  {  panelSteps.Enabled = true;  groupBoxVisualization.Enabled = true;  groupBoxStatus.Enabled = true;  groupBoxSteps.Enabled = true;  groupBoxKnapsack.Enabled = true;  UpdateVisualization(state);  buttonPrevious.Enabled = false;  }  private void UpdateVisualization(KnapsackState state)  {  visualizer.Visualize(state);  buttonNext.Enabled = true;  buttonPrevious.Enabled = true;  labelTotalSteps.Text = $"Количество шагов:\n{manager.Storage.CountOfStates}";  labelCurrentStep.Text = $"Текущий шаг:\n{state.NumStep}";  labelCapacity.Text = $"Вместимость:\n{state.Capacity}";  labelCurrentWeight.Text = $"Текущий вес:\n{state.TotalWeight}";  labelCurrentValue.Text = $"Текущая стоимость:\n{state.TotalValue}";  if (state.NumStep == manager.Storage.CountOfStates)  {  labelStatus.Text = $"Статус:\nРюкзак заполнен";  return;  }  switch (state.NumStep)  {  case 1:  labelStatus.Text = $"Статус:\nПредметы выбраны";  break;  case 2:  labelStatus.Text = $"Статус:\nПредметы\nотсортированы";  break;  default:  labelStatus.Text = $"Статус:\nРюкзак заполняется";  break;  }  }  private void ButtonNext\_Click(object sender, EventArgs e)  {  if (manager == null)  {  return;  }  KnapsackState state = manager.Storage.GetNextState();  UpdateVisualization(state);  if (state.NumStep == manager.Storage.CountOfStates)  {  buttonNext.Enabled = false;  if (checkBoxShowMessageBox.Checked)  {  MessageBox.Show("Рюкзак заполнен", "Результат", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  }  }  private void ButtonPrevious\_Click(object sender, EventArgs e)  {  if (manager == null)  {  return;  }  KnapsackState state = manager.Storage.GetPreviousState();  UpdateVisualization(state);    if (state.NumStep == 1)  {  buttonPrevious.Enabled = false;  }  }  private void InfoToolStripMenuItem\_Click(object sender, EventArgs e)  {  InformationForm infoForm = new InformationForm();  infoForm.ShowDialog();  }  private void SaveToolStripMenuItem\_Click(object sender, EventArgs e)  {  if (manager == null)  {  MessageBox.Show("Данные для сохранения отсутствуют", "Результат", MessageBoxButtons.OK, MessageBoxIcon.Error);  return;  }  if (saveFileDialog.ShowDialog() == DialogResult.OK)  {  try  {  manager.Storage.SaveToFile(saveFileDialog.FileName);  MessageBox.Show("Сохранение прошло успешно", "Результат", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  catch (IOException ex)  {  MessageBox.Show(ex.Message, "Результат", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  }  private void LoadToolStripMenuItem\_Click(object sender, EventArgs e)  {  if (openFileDialog.ShowDialog() == DialogResult.OK)  {  try  {  KnapsackParameters parameters = new KnapsackParameters(new List<Item>(), 1);  manager = new KnapsackManager(parameters);  manager.Storage.LoadFromFile(openFileDialog.FileName);  InitializeStateShowing(manager.Storage.Reset());  MessageBox.Show("Загрузка прошла успешно", "Результат", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  catch (IOException ex)  {  MessageBox.Show(ex.Message, "Результат", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  }  } |

|  |
| --- |
| namespace CourseWork;  /// <summary>  /// Форма ввода  /// </summary>  public partial class InputForm : Form  {  private List<Item> items;  public KnapsackParameters? Parameters { get; private set; }  public event EventHandler? ParametersSubmitted;  public InputForm()  {  InitializeComponent();  items = new List<Item>();  }  private void ButtonAddItem\_Click(object sender, EventArgs e)  {  int weight = (int)numericUpDownWeight.Value;  int value = (int)numericUpDownValue.Value;  items.Add(new Item(weight, value));  listBoxItems.Items.Add($"Вес: {weight}, Стоимость: {value}");  }  private void ButtonSubmit\_Click(object sender, EventArgs e)  {  if (items.Count == 0)  {  MessageBox.Show("Не введены предметы");  return;  }  int capacity = (int)numericUpDownCapacity.Value;  Parameters = new KnapsackParameters(items, capacity);  ParametersSubmitted?.Invoke(this, EventArgs.Empty);  this.Close();  }  private void ButtonReset\_Click(object sender, EventArgs e)  {  items.Clear();  listBoxItems.Items.Clear();  numericUpDownWeight.Value = 5;  numericUpDownValue.Value = 10;  numericUpDownCapacity.Value = 10;  }  } |