Для решения поставленных задач была использована реализация языка Prolog –SWI-Prolog.

**Задача №1 (Lab7-1-1.pl): (не реализовано взаимодействие с пустым списком)**

Для каждого задания было создано окно (кроме 3, так как оно не требует дополнительных данных и сразу выводит необходимый результат) для ввода списка, а также при необходимости других аргументов (задание 4: значение i).

Для того, чтобы вызвать эти окна надо в окне «Меню» (menu()) рис. 1, выбрать соответствующие заданию названия:

1. Задание 1 – «Сортировка Шелла» (shell\_win()) ;
2. Задание 2 – «Сортировка Хоара» (qsort\_win());
3. Задание 3 – «Числа по периметру» (number\_win());
4. Задание 4 – «Удалить элементы» (delete\_win());

Изображение выглядит как текст

Автоматически созданное описание

рис. 1. Окно "Меню"

В окне «Сортировка Шелла» рис. 2 необходимо ввести список для сортировки (числа через пробел «2 4 -1 1 3»). При нажатии на кнопку «Сортировать», откроется новое окно, в котором будет выведен введенный список в отсортированном (Сортировка Шелла) виде рис. 3.

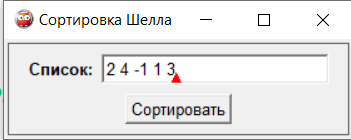


рис. 2. Окно "Сортировка Шелла".

Изображение выглядит как текст

Автоматически созданное описание

рис. 3. Отсортированный список (Сортировка Шелла)

В окне «Сортировка Хоара» рис. 4 также необходимо ввести список для сортировки (числа через пробел «4 5 2 0 42 1»). При нажатии на кнопку «Сортировать», откроется новое окно, в котором будет выведен введенный список в отсортированном (Сортировка Хоара) виде рис. 5.

Изображение выглядит как текст

Автоматически созданное описание

рис. 4. Окно "Сортировка Хоара".

Изображение выглядит как текст

Автоматически созданное описание

рис. 5. Отсортированный список (Сортировка Хоара).

Задание 3 не требует ввода каких-либо данных поэтому при нажатии на «Числа по периметру» сразу откроется окно с необходимым результатом рис. 6.

Изображение выглядит как текст

Автоматически созданное описание

рис. 6. Решение задания 3.

В окне «Удалить элементы» рис. 7 необходимо ввести список (числа через пробел «1 2 3 4 5 6 7 8 9»), а также значение i (число, например 2). При нажатии на кнопку «Изменить список», откроется новое окно, в котором будет выведен новый список, в котором будут удалены элементы, находящиеся на (i \* n) – х местах, где i задается в качестве аргумента, а в n = 1, 2, 3, … введенного списка рис. 8.

Изображение выглядит как текст

Автоматически созданное описание

рис. 7. Окно "Удалить элементы".

Изображение выглядит как текст

Автоматически созданное описание

рис. 8. Результат выполнения задания 4.

**Примеры выполнения программы при различных данных:**

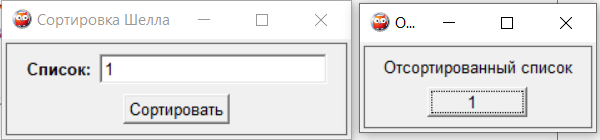


рис. 9. Сортировка Шелла. Пример 1

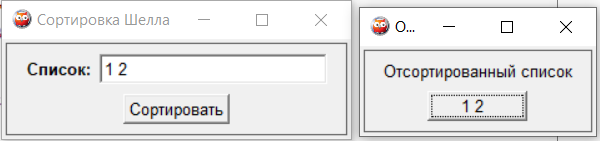


рис. 10. Сортировка Шелла. Пример 2

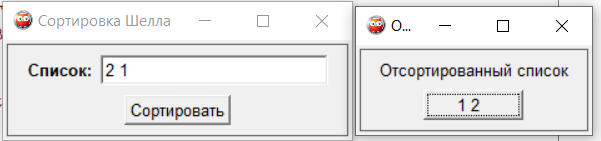


рис. 11. Сортировка Шелла. Пример 3

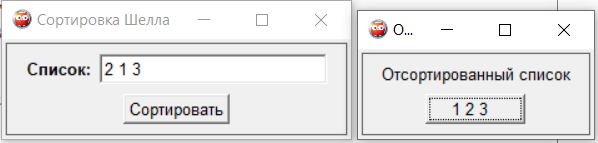


Рисунок 12. Сортировка Шелла. Пример 4

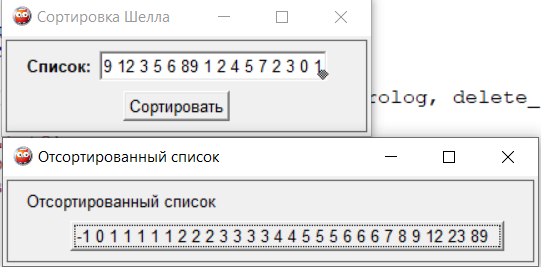


рис. 13. Сортировка Шелла. Пример 5 (Исходный список: "1 2 6 23 -1 1 3 1 3 4 5 6 8 9 12 3 5 6 89 1 2 4 5 7 2 3 0 1")



рис. 14. Сортировка Хоара. Пример 1

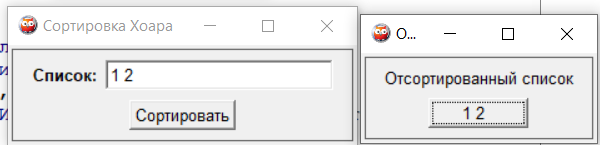


рис. 15. Сортировка Хоара. Пример 2

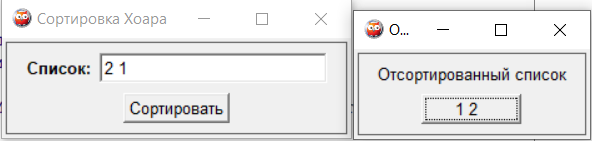


рис. 16. Сортировка Хоара. Пример 3

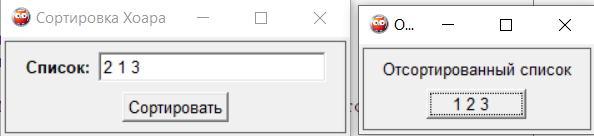


рис. 17. Сортировка Хоара. Пример 4

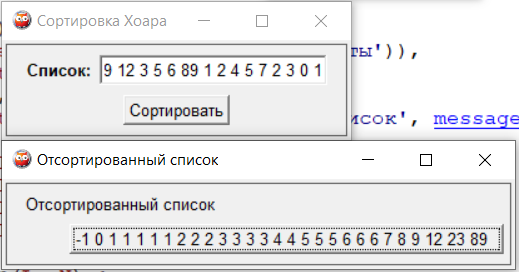


Рисунок 18. Сортировка Хоара. Пример 5 (Исходный список: "1 2 6 23 -1 1 3 1 3 4 5 6 8 9 12 3 5 6 89 1 2 4 5 7 2 3 0 1")

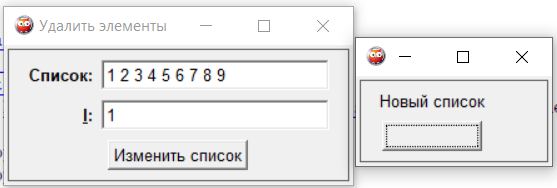


Рисунок 19. Задание 4. Пример 1

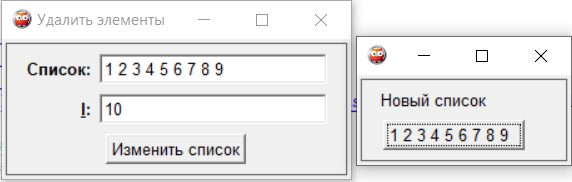


Рисунок 20. Задание 4. Пример 2

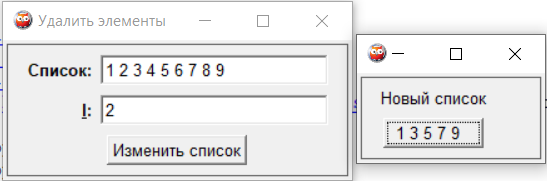


Рисунок 21. Задание 4. Пример 3

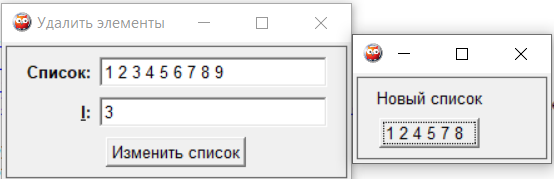


Рисунок 22. Задание 4. Пример 4

**Задача №2 (Lab7-2.pl):**

Для вывода состояния игры используется show(). 1 – крестик, 2 – нолик, 0 – не занятая позиция. Первыми ходят крестики. Для процедуры “Следующий ход”, которая для заданного положения на доске находила бы наилучший ход, обеспечивающий либо предотвращение проигрыша, либо выигрыш, либо наилучший прогнозируемый результат, были написаны правила possible\_move\_o().

1. Если центр не занят, то занимаем его.
2. Ситуации, при которых, данный ход является выигрышным, занимаем победную позицию.
3. Ситуации, в которой нужно предотвратить выигрыш противника.
4. Если центр занят, то занимает левый правый угол.
5. Если центр заняли мы - нолики, а игрок занял одну из угловых позиций, то занять позиция в углу напротив.
6. В иных случаях проверяем занятые ли угловые позиции, если свободна одна, то занимает, иначе занимаем любую свободную позицию.

Приведенный выше возможные ситуации, приведены в приоритетном порядке, поэтому в правилах possible\_move\_o() используются отсечения, так как нам необходимо получить только первое решение (наилучшее).

Для запуска программы следует ввести:

game().

**Примеры работы программы:**

**Проигрыш игрока:**

Изображение выглядит как текст

Автоматически созданное описание

рис. 23. Пример 1 (Часть 1)

Изображение выглядит как стол

Автоматически созданное описание

рис. 24. Пример 2 (Часть 2)

**Ничья:**

Изображение выглядит как текст

Автоматически созданное описание

рис 25. Пример 2 (Часть 1)

Изображение выглядит как текст

Автоматически созданное описание

рис. 26. Пример 2 (Часть 2).

Изображение выглядит как текст

Автоматически созданное описание

рис. 27. Пример 2 (Часть 3).

**Задача №3 (Lab7-3.pl):**

Для начала игры в окне «Меню» (menu()) рис. 28, необходимо нажать кнопку «Начать игру», для вызова окна игры в «Крестики-нолики» 3x3 рис. 29.

Изображение выглядит как текст

Автоматически созданное описание

рис. 28. Окно "Меню" игры крестики-нолики

Изображение выглядит как мебель, комод

Автоматически созданное описание

рис. 29. Поле игры "Крестики-нолики"

Игрок ходит крестиками. На рис. 29. левое поле, пронумерованное от 1 до 9, соответствует положению полей на правом поле, поэтому при нажатии кнопок на левом поле, на правом поле в соответствующей позиции появиться «X» рис. 30.

Изображение выглядит как мебель, снимок экрана, комод

Автоматически созданное описание

рис. 30. При нажатии на “5” в соответствующей ей позиции появился "X"

**Примеры работы программы:**

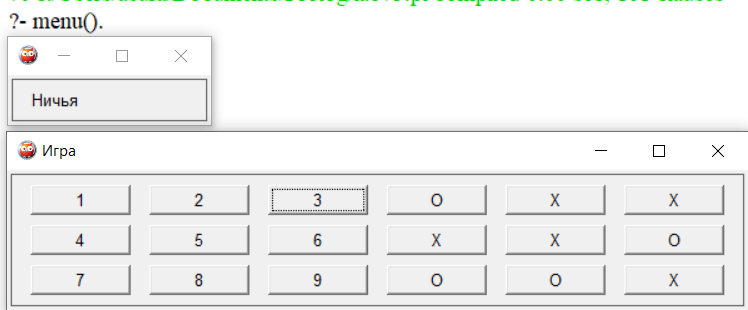


рис. 31. Ничья

Изображение выглядит как мебель, комод

Автоматически созданное описание

рис. 32. Победа ноликов.

**Задача №4 (Lab7-1-2.pl):**

Для решения задачи было создано главное окно «Меню» (menu()) рисунок 1, где реализованы 4 кнопки, которые ведут к окнам, которые решают поставленные задачи:

1. «Сортировка Шелла» (shell\_win()) - сортировка списка методом Шелла рисунок 2;
2. «Сортировка прямым выбором» (selection\_win())- сортировка методом прямого выбора рисунок 3;
3. «Задача “Числа по периметру”» (number\_result()) - решение головоломки «Числа по периметру» рисунок 4;
4. «Вставка подсписка в список» (insert\_win()) - вставка подсписка в список, начиная с i-го элемента рисунок 5.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1. Окно "Меню"

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2. Окно "Сортировка Шелла"

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3. Окно "Сортировка прямым выбором"

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4. Окно-результат задачи “Числа по периметру”.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5. Окно «Вставка подсписка в список»

Список элементы списка необходимо вводить через пробел, например, «32 5 3 6 3 2 7 5 9 6 3 5».

**Примеры выполнения программы при различных данных:**

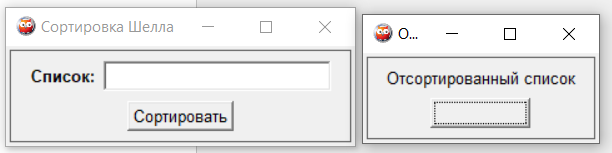


Рисунок 6. Сортировка Шелла. Пример 1

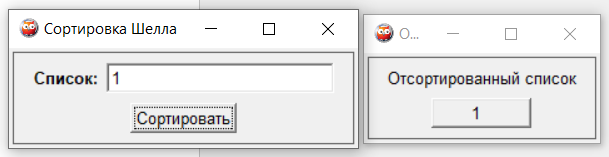


Рисунок 7. Сортировка Шелла. Пример 2

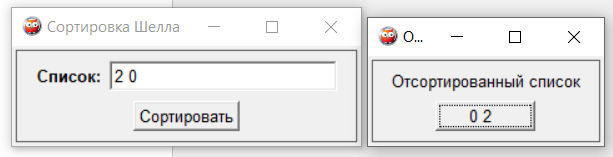


Рисунок 8. Сортировка Шелла. Пример 3

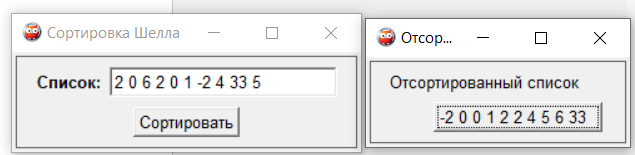


Рисунок 9. Сортировка Шелла. Пример 4

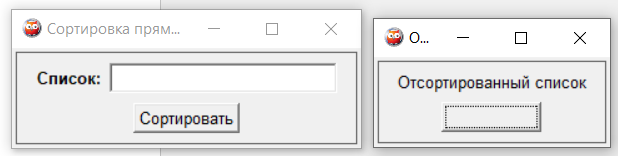


Рисунок 10. Сортировка прямым выбором. Пример 1

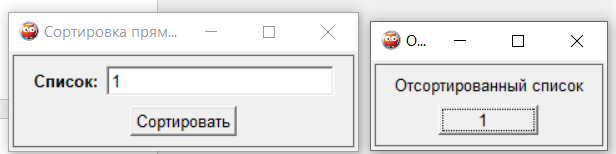


Рисунок 11. Сортировка прямым выбором. Пример 2

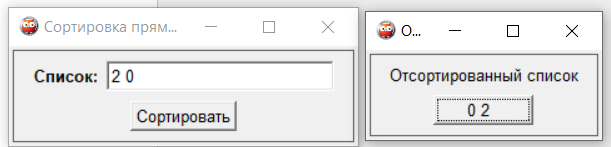


Рисунок 12. Сортировка прямым выбором. Пример 3

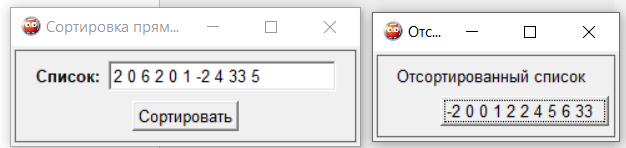


Рисунок 13. Реверсировать прямым выбором. Пример 4

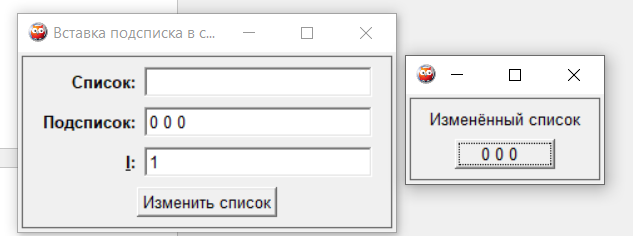


Рисунок 14. Вставка подсписка в список. Пример 1

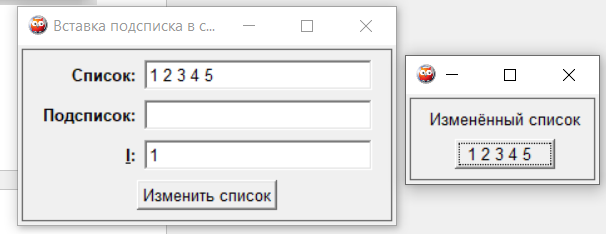


Рисунок 15. Вставка подсписка в список. Пример 2

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16. Вставка подсписка в список. Пример 3

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17. Вставка подсписка в список. Пример 4

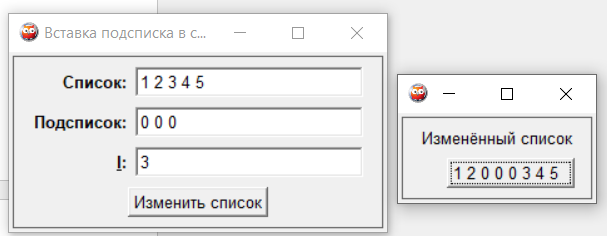


Рисунок 18. Вставка подсписка в список. Пример 5

**Задача №5 (Lab7-1-3.pl):**

Для решения задачи было создано главное окно «Меню» (menu()) рисунок 1, где реализованы 4 кнопки, которые ведут к окнам, которые решают поставленные задачи: «Сортировка Шелла» (shell\_win()) - сортировка списка методом Шелла рисунок 2; «Сортировка пузырьком» (bubble\_win())- сортировка методом пузырька рисунок 3; «Задача “Числа по периметру”» (number\_result()) - решение головоломки «Числа по периметру» рисунок 4; «Реверсировать часть списка» (revers\_win()) - преобразование списка, которое реверсирует n элементов исходного списка, начиная с i-го рисунок 5.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1. Окно "Меню"

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2. Окно "Сортировка Шелла"

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3. Окно "Сортировка пузырьком"

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4. Окно-результат задачи “Числа по периметру”.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5. Окно «Реверсировать часть списка»

Список элементы списка необходимо вводить через пробел, например, «1 5 3 2 4 5 8 5 2 5 7 9».

**Примеры выполнения программы при различных данных:**

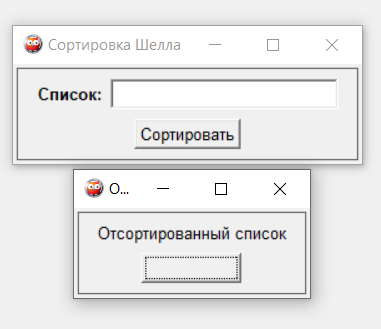


Рисунок 6. Сортировка Шелла. Пример 1

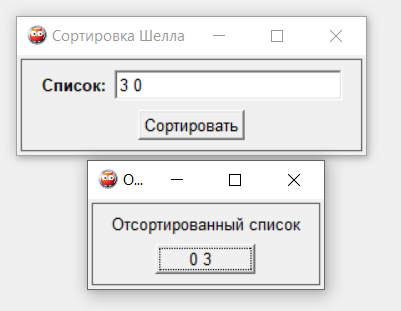


Рисунок 7. Сортировка Шелла. Пример 2

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8. Сортировка Шелла. Пример 3

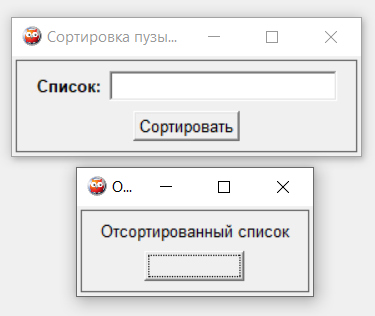


Рисунок 9. Сортировка пузырьком. Пример 1

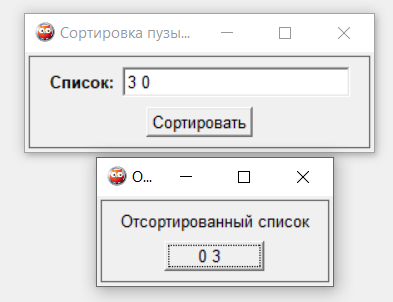


Рисунок 10. Сортировка пузырьком. Пример 2

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11. Сортировка пузырьком. Пример 3

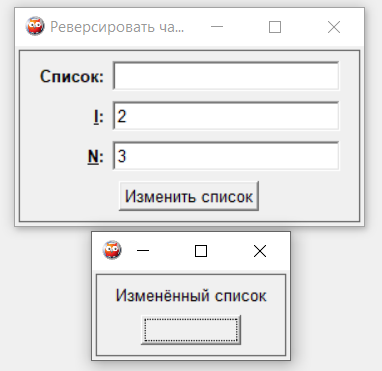


Рисунок 12. Реверсировать часть списка. Пример 1

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13. Реверсировать часть списка. Пример 2